CodeReporter 2

Benutzer- und Referenzhandbuch

Deutsche Version

Der Report-Writer für C-/C++-/BASIC-Entwickler
dBASE-kompatibel
FoxPro-kompatibel
Clipper-kompatibel

Übersetzung ins Deutsche: ComFood Software GmbH

Dieses Handbuch und die zugehörige Software werden ohne die Garantie verkauft, daß sie für einen bestimmten Zweck verwendet werden können. Dies betrifft insbesondere die kommerzielle Verwendung der Software oder des Handbuchs.

Die Informationen in diesem Handbuch und die Funktion der zugehörigen Software können jederzeit ohne Ankündigung verändert werden. Die ComFood Software GmbH oder Sequiter Software Inc. sind nicht verpflichtet, dritten Personen evtl. Änderungen mitzuteilen.

Dieses Handbuch ist mit aller gebotenen Sorgfalt hergestellt und übersetzt worden. Trotzdem sind Weiterentwicklungen jederzeit möglich und können in die Software eingebaut werden. Abweichungen der tatsächlichen Funktion der Software von der Beschreibung im Handbuch sind daher keine Grundlage für Ansprüche gegen Sequiter Software Inc. oder die ComFood Software GmbH.

Warenzeichen

CodeBase™, CodeBasic™ und CodeReporter™ sind Warenzeichen von Sequiter Software Inc.

Borland C++® und Turbo C++® sind eingetragene Warenzeichen von Borland International.

Clipper® ist ein eingetragenes Warenzeichen von Computer Associates International Inc.

dBASE® ist ein eingetragenes Warenzeichen von Borland International.

FoxPro® ist ein eingetragenes Warenzeichen der Microsoft Corporation.

MetaWare High C™ ist ein Warenzeichen von MetaWare Inc.

Microsoft C® ist ein eingetragenes Warenzeichen der Microsoft Corporation.

Microsoft Windows® ist ein eingetragenes Warenzeichen der Microsoft Corporation.

OS/2® ist ein eingetragenes Warenzeichen der International Business Machines Corporation.

Inhaltsverzeichnis

Einleitung

CodeReporter ist ein umfassendes Programm zur Erstellung relationaler Reports, das das Entwickeln von auf Ihre Bedürfnisse zugeschnittenen Reports erleichtert.

Mit CodeReporter entwerfen Sie über einfache Auswahlbefehle ausgefeilte Reports direkt am Bildschirm. Die Daten für den Report können aus jeder beliebigen Datendatei auf Ihrem PC bzw. einer Netzwerkstation stammen. Die im Report verwendeten Datendateien werden über einfache dBASE-Ausdrücke verknüpft. Auch wenn Sie lediglich einen Teil der Datendatei, z. B. bei der Durchführung einer Abfrage, verwenden, nimmt sich CodeReporter der Schwierigkeiten an.

Die Geschwindigkeit der bitoptimierten Abfragetechnik von Sequiter Software kommt überall in CodeReporter deutlich zum Tragen. Sortier- und Abfragevorgänge sowie das Herstellen von Relationen erfolgen unübertroffen schnell. CodeReporter ist in der Lage, eine Datendatei mit 500 000 Datensätzen abzufragen und nur eine Sekunde später mit der Ausgabe zu beginnen.

Die mit CodeReporter unter Windows 3.1 entworfenen Reports sind portier- und konfigurierbar. Ein und derselbe Report läßt sich durch das Laden einer Reportdatei in verschiedenen Anwendungen benutzen; darüber hinaus können Sie Quellcode generieren, der unmittelbar in DOS-, OS/2-, UNIX- und/oder Windows-Anwendungen eingebunden werden kann. Ist die Struktur des Reports erst einmal vorhanden (fest oder variabel codiert), kann sie mit den API-Funktionen von CodeReporter so verändert werden, daß eine absolut maßgeschneiderte Lösung dabei herauskommt.

Bei der Ausgabe von Reports macht sich CodeReporter die Bildschirmdarstellung und Druckertreiber von Windows 3.1 zunutze. Jeder Teil des Reports kann nach Schriftgröße, Schriftart und/oder Farbe variiert werden. Unter Verwendung der Windows-3.1-Standard- und -TrueType-Schriften ist ein gedruckter Report mit der Darstellung auf dem Bildschirm identisch.

Diese und andere Eigenschaften machen CodeReporter zu einem unverzichtbaren Werkzeug für Software-Entwickler und Endanwender!

Eigenschaften

CodeReporter vereint die Eigenschaften zahlreicher DOS-Reportprogramme, die Vorteile von Windows 3.1 sowie die langjährigen Erfahrungen von Sequiter Software mit der Datenbankprogrammierung in sich.

Sie können eine beliebige Anzahl von Datendateien in Relation setzen und sich dabei einer der folgenden Methoden bedienen:

- Einer zu einem. Die Masterdatei enthält lediglich einen Datensatz, der auf einen Datensatz der Slavedatei paßt.
- Viele zu einem. Ein eindeutiger Suchwert aus einer Datendatei bezieht sich auf eine Vielzahl von Datensätzen in einer verknüpften Datendatei.
- Viele zu vielen. Eine beliebige Anzahl doppelter Suchwerte aus einer Datendatei bezieht sich auf eine Vielzahl von Datensätzen in einer verknüpften Datendatei.

Die Relationen lassen sich untereinander beliebig verknüpfen, so daß selbst eine Einer-zu-einem-zu-vielen-zu-vielen-zu-einem-Relation unterstützt wird.

Die Menge der in einem Report verwendeten Datensätze läßt sich mit Hilfe einfacher oder komplexer dBASE-Ausdrücke, sog. Abfragen einschränken; dabei können Felder aus allen in der Relation verwendeten Datendateien einbezogen werden. Ist Datendatei „A“ z. B. mit Datendatei „B“ verknüpft, die wiederum mit Datendatei „C“ verknüpft ist, und enthält „C“ einen Wert, der nicht in den Report eingehen soll, könnten Sie den Ausdruck **`C->FELD_NAME<>'WERT'`** verwenden.

Die in einem Report enthaltenen Zahlen sind meistens in zusammengefaßter Form am nützlichsten; lange Zahlenkolonnen gewinnen erst mit der letzten Zeile Sinn. CodeReporter ist in der Lage, numerische Daten auf verschiedene Art und Weise zusammenzufassen:

- Sie können die Summe einer Gruppe von Zahlen ermitteln lassen.
- Sie können den höchsten bzw. niedrigsten auftretenden Wert speichern.
- Sie können einen Mittelwert errechnen lassen.

Darüber hinaus lassen sich Summen als „Vorschau“-Summen definieren. Das heißt, die Summe überspringt die eigentliche Reportausgabe und setzt ihre Summierung fort; auf diese Weise werden Summenwerte innerhalb des Reports ermittelt und ausgegeben, bevor diese Informationen angezeigt werden. Diese Summe läßt sich z. B. in späteren Berechnungen verwenden, um den Prozentwert einer Summe anzuzeigen.

CodeReporter benutzt die unter Windows 3.1 verfügbaren Schriften, einschließ-

lich der neuen TrueType-Fonts. Über die Erstellung eines Styles können Sie eine beliebige Anzahl verschiedener Schriften, Schriftgrößen und Farben in einem Report verwenden. Alle Schriften, die Windows darstellen kann, können Sie in einem Report benutzen.

Innerhalb des Reports können Sie beliebige Ausgabeobjekte mittels Mausbefehl verschieben. Darüber hinaus lassen sich diese Objekte mit den Befehlen „Ausschneiden“ und „Kopieren“ auf einfache Weise vervielfältigen.

In der Entwurfsphase eines Reports läßt sich eine Menge Papier sparen, wenn Sie sich die Ausgabe probehalber auf dem Bildschirm anzeigen lassen. Da CodeReporter Windows-Fonts benutzt, stimmen Papierausdruck und Bildschirmdarstellung weitgehend überein.

Haben Sie mit CodeReporter einen Report entwickelt, können Sie ihn als variabel codierte Reportdatei für die nächste CodeReporter-Sitzung bzw. als Quellcode sichern. Der generierte Code läßt sich in alle Anwendungen einbinden, die mit einem Datenbankprogramm von Sequiter Software und dem CodeReporter- API arbeiten, und zwar unter DOS, OS/2 oder Windows.

Lockern Sie Ihre Reports mit Grafiken auf! Dazu können Sie entweder statische Grafikelemente wie Firmenlogos einbauen oder solche Elemente unmittelbar in den ablaufenden Report laden.

CodeReporter 2.0 ist in der Lage, einen Report in eine Datendatei auszugeben. Diese neue Eigenschaft gestattet es dem Benutzer, einen Report auf Grundlage eines bereits vorhandenen Reports zu erstellen bzw. CodeReporter als Datenumformungswerkzeug einzusetzen.

Für Programmentwickler stehen preisgünstige Mehrfachlizenzen von CodeReporter zur Verfügung, die an Endanwender verkauft werden können. Einzelheiten erfragen Sie bitte telefonisch oder schriftlich.

Starten mit CodeReporter

Systemanforderungen

Die mit CodeReporter und den Reportfunktionen erstellten Reports können unter jedem beliebigen Betriebssystem eingesetzt werden. CodeReporter selbst jedoch läuft nur unter Windows 3.1.

Hinweis: Zur Bedienung von CodeReporter benötigen Sie eine Windows-kompatible Maus.

Um die Funktionen von CodeReporter voll ausnutzen zu können, empfehlen wir die folgende Minimalkonfiguration:

- IBM-386er oder 100 % kompatibler Rechner mit einer Taktfrequenz von mindestens 25 MHz,
- mindestens 4 MB Hauptspeicher,
- Windows-kompatibler SVGA-Farbmonitor,
- Windows-kompatible Maus mit zwei Tasten,
- eine Festplatte mit mindestens 1,6 MB Plattenplatz für jede installierte CodeReporter-Version (Je nach Report kann der Programmlauf zusätzliche Plattenkapazität erforderlich machen.).

CodeReporter läuft auf nahezu jedem Computersystem, auf dem Windows 3.1 installiert ist; allerdings hängt die Schnelligkeit der Reporterzeugung von der Leistungsfähigkeit des Systems ab.

Registrierung

Bitte nehmen Sie sich einen Augenblick Zeit, um die CodeReporter-Registrierkarte auszufüllen und an uns zu senden. Nur so stellen Sie für sich rasche technische Unterstützung und den Update-Service sicher.

Installation

Haben Sie CodeReporter als Teil eines Sequiter-Programmpakets erworben, erfolgt die Installation über das mitgelieferte Installationsprogramm. Kaufen Sie CodeReporter getrennt, wird die Installation über die Datei INSTALL.EXE auf Programmdiskette 1 ausgeführt.

Versionen

Aufgrund einer indexspezifischen CodeBase-DLL kann CodeReporter mit jedem der vier unterstützten Dateiformate eingesetzt werden. Beim Aufruf sucht das Programm die Bibliothek CR2CODE.DLL, um deren Indexdatei-Kompatibilität festzustellen.

Erwerben Sie CodeReporter getrennt von CodeBase, legt das Installationsprogramm die entsprechende indexdateispezifische DLL im CodeReporter-Verzeichnis ab. Wünschen Sie eine andere Indexversion, müssen Sie CodeReporter neu installieren.

Wird CodeReporter als Teil eines Programmpakets (z. B. CodeBase 5.1 & CodeReporter) installiert, wird die gewünschte Indexversion von CR2CODE.DLL im CodeReporter-Verzeichnis abgelegt. Wie man eine andere Indexversion installiert, entnehmen Sie bitte der Dokumentation des Programmpakets.

CodeReporter unterstützt folgende Indexdateiformate:

- dBASE-IV-Indexdateien (.MDX). Wenn Sie dBASE-IV-Indexdateien verwenden, können Sie nicht gleichzeitig mit dBASE-III-PLUS-Indizes (.NDX) arbeiten, auch wenn das mit dBASE IV möglich ist.
- Kompakte Indexdateien (.CDX und .IDX) von FoxPro 2.0 (oder höher). CodeReporter unterstützt die älteren nichtkompakten Indexdateien von FoxPro nicht.
- Clipper-Indexdateien (.NTX). CodeReporter arbeitet sowohl mit Clipper Sommer '87 als auch mit Clipper 5.
- dBASE-III-PLUS-Indexdateien (.NDX).

CodeReporter aufrufen

Da es sich bei CodeReporter um eine Windows-Anwendung handelt, muß Microsoft Windows 3.1 (bzw. eine höhere Version) ordnungsgemäß installiert und geladen sein, ehe Sie CodeReporter aufrufen können.

Sie können CodeReporter von der DOS-Befehlszeile aus aufrufen, indem Sie **WIN** und der Namen der CodeReporter-Version eingeben, die ausgeführt werden soll. Die FoxPro-Version z. B. würden Sie mit **WIN CREP2** vom CodeReporter-Stammverzeichnis **C:\CODEREP** aus aufrufen. Da Windows-Anwendungen im allgemeinen jedoch nicht von der DOS-Befehlszeile aus aufgerufen werden, sind folgende Möglichkeiten vorzuziehen.

Datei-Manager

Rufen Sie, nachdem Sie Windwos gestartet haben, den Datei-Manager durch Doppelklicken auf das Symbol (das sich im allgemeinen in der „Hauptgruppe“ befindet) auf. Wählen Sie mit dem Datei-Manager das Laufwerk und das Verzeichnis an, in dem CodeReporter installiert wurde (Standard: **C:\CODEREP**). Wird die Dateiliste für das CodeReporter-Verzeichnis angezeigt, suchen Sie darin den Dateinamen der ausführbaren CodeReporter-Version (z. B. **CREP2.EXE**).

Wenn Sie auf diesem Dateinamen doppelklicken, wird CodeReporter gestartet.

Datei | Ausführen

Daneben kann CodeReporter über die Menüoptionen des Windows-Programm-Managers aufgerufen werden. Klicken Sie dazu den Menüpunkt Datei | Ausführen an. Windows zeigt eine Dialogbox an, in die Sie Laufwerk, Pfad und Programmnamen von CodeReporter eingeben können.

Haben Sie CodeReporter auf Laufwerk C: im Standard-CodeReporter-Verzeichnis installiert, würde die Clipper-kompatible Version mit folgender Eingabe in die Dialogbox aufgerufen:

C:\CODEREP\CREP2.EXE

CodeReporter unterstützt einen Befehlszeilenparameter, der als Name einer Reportdatei interpretiert wird. Bei Aufruf mit Befehlszeilenparameter wird der angegebene Report automatisch geladen. Der Aufruf ohne Befehlszeilenparameter startet lediglich CodeReporter.

Symbol

Unter Windows werden Symbole verwendet, damit der Benutzer rasch auf die entsprechenden Anwendungen zugreifen kann. Das Symbol für CodeReporter kann zu jeder beliebigen Programmgruppe hinzugefügt und, wie andere Windows-Anwendungen auch, durch Doppelklicken aktiviert werden. Das ist die einfachste Methode, CodeReporter aufzurufen.

Weitere Einzelheiten zum Anlegen von Programmgruppen und Hinzufügen von Symbolen entnehmen Sie bitte Ihrem Windows-Benutzerhandbuch.

Zugriff auf Reportdateien

Die mit CodeReporter erzeugten Reports können als Reportdateien (Erweiterung am Dateinamen „.REP“) abgespeichert werden. Diese Dateien enthalten sämtliche reportspezifischen Angaben, einschließlich der Pfade zu den Datendateien der Relationen und der verwendeten Styles.

Einen Report laden

Ein bereits vorhandener Report kann über die Menüoption **DATEI | ÖFFNEN** erneut geladen werden. Geben Sie im Dialog „Datei öffnen“ die gewünschte Datei an, und klicken Sie auf „OK“.

Hinweis: Versuchen Sie, bei bereits geladener Reportdatei eine weitere Reportdatei zu laden, schließt CodeReporter den aktuellen Report.

Sie können eine Reportdatei auch mit der Option **DATEI | ÖFFNEN MIT PFAD** laden. Neben dem Dateinamen bittet die Option **DATEI | ÖFFNEN MIT PFAD** Sie ebenfalls um die Eingabe des Verzeichnisses, in dem sich die Datendateien des Reports befinden.

Hinweis: Die Option **DATEI | ÖFFNEN MIT PFAD** sollten Sie verwenden, wenn die Datendateien eines Reports in ein anderes Verzeichnis kopiert worden sind.

Über die Menüoption **DATEI | ALTE DATEI ÖFFNEN** lassen sich Dateien von CodeReporter 1.0 in CodeReporter 2.0 importieren.

Einen neuen Report anlegen

Mit der Menüoption **DATEI | NEU** können Sie einen neuen Report erzeugen. Der aktuelle Report wird geschlossen, und der Dialog „Datei öffnen" wird aufgerufen, damit Sie die Top-Master-Datendatei angeben können.

Automatisch erzeugt CodeReporter eine neue Gruppe namens „BODY" und zeigt diese in dem neuen Report an. Der Name des Reports wird beim Abspeichern der Datei angegeben.

Einen Report speichern

Die Option **DATEI | SPEICHERN** veranlaßt CodeReporter, den geänderten Report auf die Platte zu schreiben. Wurde ein Report bislang noch nicht gesichert (z. B. ein neuer), ruft CodeReporter den Dialog „Datei öffnen" auf, damit Sie einen Namen für den Report eingeben.

Mit der Option **DATEI | SPEICHERN UNTER** kann ein Report unter einem neuen Namen abgelegt werden. Hierbei fordert CodeReporter Sie, wie bei einem neuen Report, zur Eingabe eines Dateinamens auf.

Drucktechnische Vereinbarungen

Folgender Konventionen bedienen wir uns im vorliegenden Handbuch:

Die Menüoptionen werden in fettgedruckten Kapitälchen dargestellt; ist ein Untermenü vorhanden, folgt dem Haupteintrag ein „|" sowie der entsprechende Name. Darüber hinaus leiten wir die Menübezeichnung im allgemeinen mit dem Begriff „Menüoption" bzw. „Option" ein.

Beispiel: Wählen Sie die Menüoption **DATEI | NEU** an.

Die Bezeichnungen von Dialogboxen sind in doppelte Anführungszeichen eingeschlossen und erscheinen in normaler Schrift. Darüber hinaus werden dem Namen der Dialogbox im allgemeinen die Begriffe „Dialogbox" oder „Dialog" vorangestellt.

Beispiel: Verwenden Sie die Dialogbox „Ausdruck eingeben" für die Eingabe von dBASE-Ausdrücken.

Bezeichnungen von Befehlsschaltflächen in Dialogboxen oder auf dem Code-Reporter-Entwicklungsbildschirm sind in doppelte Anführungszeichen eingeschlossen und erscheinen in normaler Schrift. Darüber hinaus wird der Typ der Schaltfläche im allgemeinen ihrer Bezeichnung vorangestellt.

Beispiel: Wählen Sie einen Eintrag in der Listbox „Felder“ an, und klicken Sie auf „OK“.

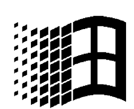
dBASE-Ausdrücke erscheinen in fetter Schreibmaschinenschrift und Großbuchstaben (die einzige Ausnahme dabei bildet der zu einen dBASE-Ausdruck gehörende Text). Die in einem solchen Ausdruck genannten Felder werden mit dem Namen der Datendatei, zu der sie gehören, und einem „->“ bezeichnet. Hinweise zu den dBASE-Ausdrücken finden Sie im Kapitel „Ausdrücke“.

Beispiel: **`'Name: '+ DATA->NAME+'Alter: '+STR(DATA->ALTER,3,0)`**

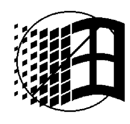
Bei allen in diesem Handbuch genannten Verzeichnissen gehen wir davon aus, daß es sich um Unterverzeichnisse des CodeReporter-Verzeichnisses handelt (Standard: C:\CODEREP). Bei der Nennung von Unterverzeichnissen wird das CodeReporter-Verzeichnis als „.\“ aufgeführt, was so viel wie das „aktuelle Verzeichnis“ bedeutet. .\EXAMPLES bzw. C:\CODEREP\EXAMPLES beziehen sich also auf ein und dasselbe Verzeichnis.

Symbole

Im vorliegenden Handbuch benutzen wir folgende Symbole, wenn es darum geht, auf wichtige Informationen hinzuweisen.



Dieses Symbol zeigt an, daß sich die entsprechende Funktion auf Windows-Anwendungen beschränkt. Sie ist lediglich dann verfügbar, wenn die Reportmodulfunktionen mit der Option **S4WINDOWS** kompiliert worden sind. Das Symbol enthält eine Grafik, deren Rechte bei der Microsoft Corporation liegen.



Das nebenstehende Symbol zeigt an, daß die entsprechende Funktion nicht in Windows-Anwendungen verwendet werden kann. Das Symbol enthält eine Grafik, deren Rechte bei der Microsoft Corporation liegen.

Der Report-Entwicklungsbildschirm

An verschiedenen Stellen in diesem Handbuch werden Teile des CodeReporter-Entwicklungsbildschirms angesprochen, wie sie aus der folgenden Abbildung hervorgehen.

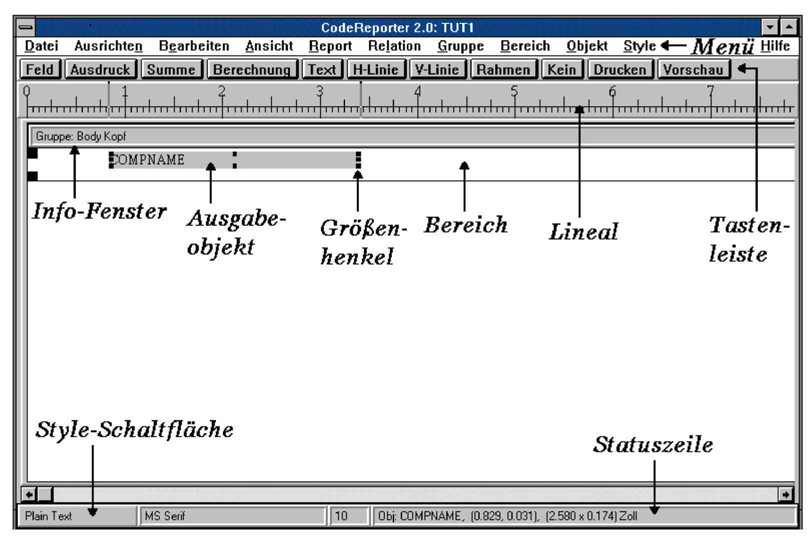


Abb. 1: Report-Entwicklungsbildschirm

Danke für Ihre Mithilfe

Sollten Sie Fragen oder Anregungen zu diesem bzw. anderen Produkten von Sequiter haben, wenden Sie sich bitte per Telefon, Modem, Fax oder Brief an uns. Ihre Meinung und Ihre Fragen sind uns sehr wichtig. Wenn Sie mit der Qualität des Produkts oder unserem Service nicht vollauf zufrieden sind, wenden Sie sich mit Ihrem Problem einfach an unsere „Qualitätssicherung“.

Technische Unterstützung

Damit wir Ihnen bei Schwierigkeiten rasch helfen können, sollten Sie unserem Support-Personal folgende Angaben machen können:

1. Name, Telefon-/Telefaxnummer, Name Ihrer Firma.
2. Die Tatsache, daß Sie mit CodeReporter 2.0 arbeiten.
3. Die Seriennummer von CodeReporter. Diese finden Sie entweder auf den Programmdisketten oder vorne in diesem Handbuch.
4. Das Betriebssystem mit genauer Versionsnummer.
5. Die Dateikompatibilität der CodeReporter-Version, mit der Sie arbeiten (FoxPro, dBASE IV, dBASE III, Clipper).
6. Das Datum der Dateien auf den Programmdisketten. Von Zeit zu Zeit erscheinen neue Versionen von CodeReporter. Die Version, mit der Sie arbeiten, kann dann anhand des Datums auf den Programmdisketten exakt ermittelt werden. Wenn Sie sich den Inhalt der Disketten anzeigen lassen, wird ebenfalls das Datum mit ausgegeben.
7. Die Größe der CodeReporter-Programmdatei, mit der Sie arbeiten.

Sollen wir einen Report für Sie testen, der nicht richtig funktioniert, benötigen wir neben den obigen Angaben:

a. Eine Kopie des Reports (entweder als Nachricht in unserer Mailbox oder auf Diskette). Die in dem Report verwendeten Daten-, Index- und Memodateien sind ebenfalls erforderlich.
b. Speichern Sie die Reportdatei ohne Pfadnamen ab. (Wie man das macht, entnehmen Sie bitte dem Abschnitt „Pfadnamen“ im Kapitel „Reports gestalten“.

c. Beschreiben Sie das Problem im einzelnen.

Wir sind da, Ihnen weiterzuhelfen, und bemühen uns, Ihnen einen erstklassigen Service zu bieten.

BBS-Unterstützung

Sequiter betreibt zur Zeit seine eigene elektronische Mailbox, die technische Unterstützung und Programm-Updates bietet. Darüber hinaus nimmt Sequiter an einer Reihe nationaler und internationaler Mailboxen teil, die dieselbe technische Unterstützung bieten. Die Datei „$READ.ME“ enthält eine Liste der zur Zeit unterstützten Mailboxen sowie Hinweise zur Kontaktaufnahme mit der Technischen Unterstützung.

Adressen

Es folgen die Adressen der ausländischen Sequiter-Niederlassungen:

Sequiter Software Inc.
58–60 Beresford St.
London SE18 6BG
Großbritannien

Tel. 44 81 317 4321
Fax 44 81 317 4040

ComFood Software GmbH
Am Rohrbusch 79
D-48161 Münster
Deutschland

Tel. 0 25 34/70 93
Fax 0 25 34/88 52
BBS 0 25 34/16 63

Übungen

Das vorliegende Kapitel ist dazu gedacht, Anwender von CodeReporter 2.0 rasch mit den Grundverfahren der Reporterstellung vertraut zu machen. Nach dem Durcharbeiten dieser Übungen kann ein CodeReporter-Anwender

- vorhandene Reportdateien mit und ohne Pfad öffnen
  neue Reports aufbauen
- Text-, Ausdrucks- und Berechnungsausgabeobjekte positionieren
- Summenausgabeobjekte anlegen und positionieren
- Reportbereiche definieren und deren Größe verändern
- eine Relation manuell erzeugen bzw. von der Platte laden
- einen Report auf dem Bildschirm ausgeben
- die Optionen „Kopf austauschen“ und „Fuß austauschen“ einsetzen

Zusätzliche Beispiele zu den besonderen Eigenschaften von CodeReporter 2.0 finden sich jeweils am Ende der Kapitel „Relationale Reports“, „Bereiche“, „Ausgabeobjekte“, „Spaltenreport-Utility“, „Styles“ und „Drucken“. Die genauen Seitenzahlen entnehmen Sie bitte dem Inhaltsverzeichnis.

Einen Report mit Pfad laden

In der ersten Übung erläutern wir, wie man Reports laden und auf welche Weise man deren Datendateien lokalisieren kann.

Report im aktuellen Verzeichnis öffnen

Der erste Beispielreport, TUT1.REP, zeigt lediglich den Inhalt der Datendatei COMPANY.DBF an. Beim Abspeichern dieses Reports wurden Laufwerk und Verzeichnis der Datendatei nicht mitabgelegt. (Wie man das macht, entnehmen Sie bitte dem Kapitel „CodeReporter-Optionen“.) Daher geht das Programm davon aus, daß sich die Datendateien im selben Verzeichnis befinden wie die Reportdatei. Der Vorteil hierbei besteht darin, daß Report- und Datendateien in jedes beliebige Verzeichnis auf jedem beliebigen Laufwerk verschoben werden können; sie müssen lediglich zusammenbleiben.

Rufen Sie CodeReporter mit einer der im Kapitel „Starten mit CodeReporter" beschriebenen Methoden auf. Verwenden Sie danach die Menüoption **DATEI | ÖFFNEN**, um die Dialogbox „Reportdatei auswählen" zu aktivieren. Wählen Sie in der Listbox das Verzeichnis .\EXAMPLES an, darauf die Reportdatei TUT1.REP, und klicken Sie auf „OK". Der Report wird im aktuellen Verzeichnis geladen.

Report mit Pfad öffnen

Bei TUT2.REP handelt es sich um das genaue Gegenteil von TUT1.REP. Die verschiedenen Pfadnamen für die Datendateien COMPANY.DBF und STORES.DBF wurden innerhalb des Reports gespeichert.

Wenn Sie versuchen, diesen Report mit **DATEI | ÖFFNEN** zu laden, teilt CodeReporter Ihnen mit, daß die Datei nicht gefunden werden konnte, und fragt, ob sie für jede nicht lokalisierbare Datei eine andere Datei angeben möchten. Dieses Verfahren ist immer dann sinnvoll, wenn sich die Dateinamen oder die Verzeichnisse der im Report enthaltenen Datendateien geändert haben – die manuelle Anpassung jeder einzelnen Datei kann sich allerdings zu einem zeitraubenden Unterfangen entwickeln, insbesondere dann, wenn sich alle Dateien im selben Verzeichnis befinden.

Die Menüoption **DATEI | ÖFFNEN MIT PFAD** setzt alle Laufwerks- und Pfadangaben des Reports außer Kraft und ersetzt diese durch die des Reportentwicklers. Laden Sie die Datei TUT2.REP mit **DATEI | ÖFFNEN MIT PFAD**, und geben Sie das CodeReporter-Beispieleverzeichnis .\EXAMPLES an.

Eine Datenbankliste ausgeben

Die folgende Übung erläutert, wie man einen einfachen Report erstellt, der den Inhalt der Datendatei COMPANY.DBF ausgibt. Die Datendatei, die die Felder COMPID, COMPNAME und CEO enthält, befindet sich im CodeReporter-Verzeichnis .\EXAMPLES.

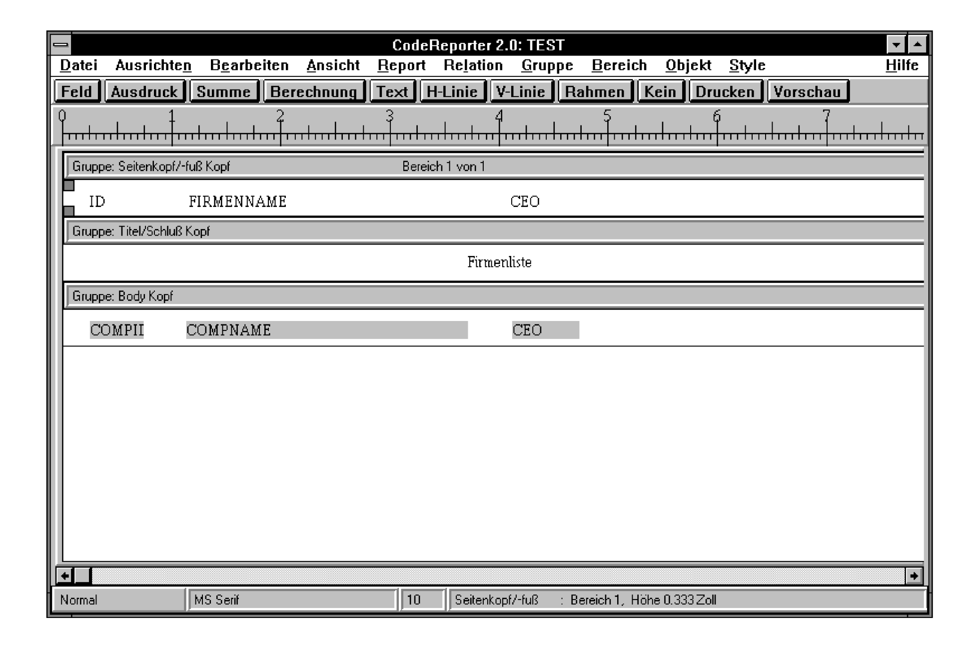


Abb. 2: Fertiggestellter Report von Übung Nr. 3

Einen neuen Report anlegen

Aktivieren Sie nach dem Aufruf von CodeReporter die Option **DATEI | NEU**. Der aktuelle Report wird geschlossen, und das Programm fordert die erste Datendatei des Reports an (die Top-Master-Datendatei). Da der Report den Inhalt der Datendatei COMPANY.DBF ausgeben soll, markieren Sie im CodeReporter-Beispieleverzeichnis die Datei COMPANY.DBF und klicken auf „OK“.

Nach Erreichen dieser Ausgangslage zeigt CodeReporter einen leeren Report-Entwicklungsbildschirm an. Es wird eine voreingestellte Gruppe namens „Body“ mit einem Kopfbereich angelegt, in den Ausgabeobjekte eingefügt werden können.

Für einen Report reicht es grundsätzlich aus, daß lediglich ein Reportbereich vorhanden ist, der für jeden zusammengesetzten Datensatz wiederholt wird. Diesen Bereich legt CodeReporter automatisch für jeden neuen Report an, so daß keine neuen Gruppen erzeugt werden müssen. (Einzelheiten zu „Gruppen“ und „Bereichen“ finden Sie in den entsprechenden Kapiteln.)

Sich wiederholende Elemente einfügen

Der voreingestellte Bereich sollte Felder aus der Top-Master-Datendatei enthalten, deren Werte sich mit jedem neuen zusammengesetzten Datensatz ändern. Um diese Felder in den Bereich „Body“ einzufügen, klicken Sie in der Tastenleiste auf die Taste „Feld“. Jetzt wird die Listbox „Feldobjekte“ (Abb. 3, unten) geöffnet, die alle Felder der zusammengesetzten Datendatei angezeigt.



Abb. 3: Listbox der Feldobjekte

Markieren Sie über die Tastatur bzw. mit der Maus, wie unter Windows üblich, alle Felder der Listbox (ziehen Sie den Cursor bei gedrückter linker Maustaste über alle drei Felder).

Nach Auswahl der gewünschten Felder positionieren Sie den Mauszeiger im Kopfbereich der Gruppe „Body“, aber klicken Sie zu diesem Zeitpunkt noch nicht auf die Schaltfläche „Fertig“. Aus dem Pfeil-Cursor wird ein Feldeinfügecursor (die einzelnen Cursorformen finden Sie in Anhang C). Das nächste Klikken der linken Maustaste bestimmt die Position, der linken obere Ecke der Felder.

Bewegen Sie das Kreuz des Feldeinfügecursors links in den Bereich „Body“, und drücken Sie die linke Maustaste; die Dialogbox „Feldgestaltung“ wird aufgerufen. Da die Voreinstellungen in Ordnung sind, brauchen Sie nur noch auf „OK“ zu klicken, um die Positionierung der Felder zu beenden. Der Report-Entwicklungsbildschirm hat nun folgendes Aussehen (siehe nächste Seite).

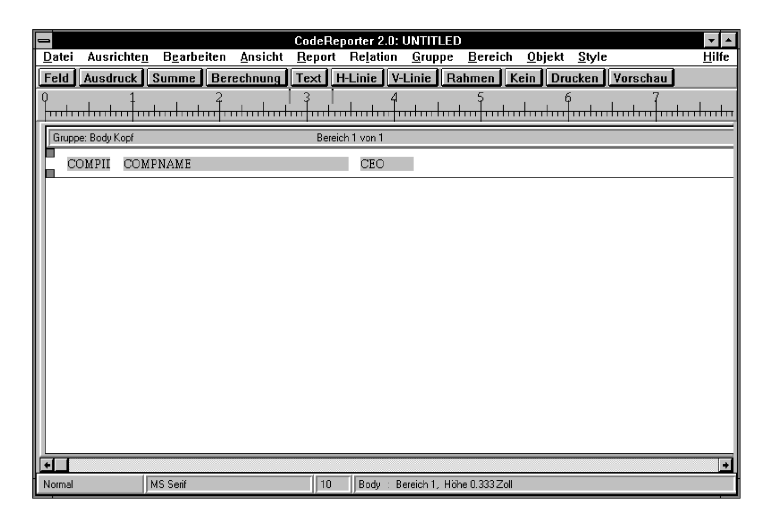


Abb. 4: Noch nicht fertiggestellter Report von Übung Nr. 3

Report einsehen

Eigentlich können wir den Report an dieser Stelle als fertiggestellt betrachten, da er seinen Zweck, die Ausgabe des Inhalts der Datendatei COMPANY.DBF, erfüllt hat.

Mit der Menüoption **DATEI | DRUCKBILD EINSEHEN** können Sie sich den Report anzeigen lassen. Wie Sie in Abb. 5 sehen, handelt es sich um einen ziemlich langweiligen und relativ nichtssagenden Report. Ist ein Leser nicht mit dem Inhalt und Aufbau aller verfügbaren Datendateien vertraut, kann er unmöglich wissen, was dieser Report darstellen soll.

Bereiche zur Erläuterung einfügen

Ein geeigneter Zusatz zur Erläuterung des Reports wäre ein Titel wie „Inhalt von COMPANY.DBF“. Es ist jedoch wenig sinnvoll, diesen Titel in den Kopfbereich der Gruppe „Body“ zu setzen, da er dann bei jedem Datensatz der zusammengesetzten Datendatei wiederholt wird.

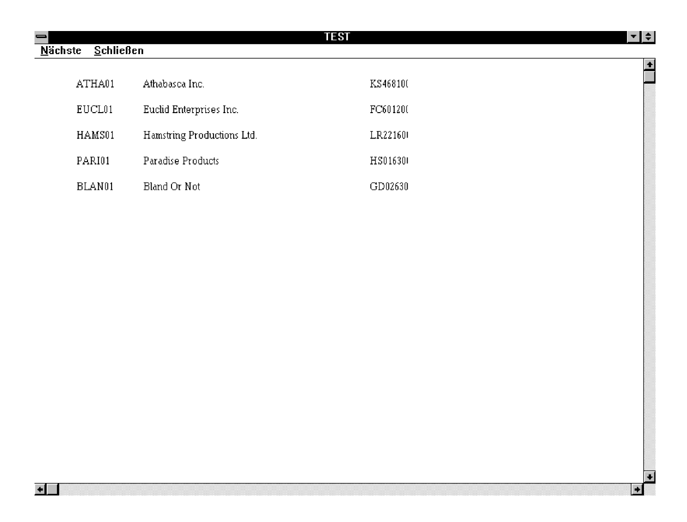


Abb. 5: Teilausgabe von Übung Nr. 3

Eine Überschrift für den Report gehört in den Titelbereich. Alle Elemente darin werden oben auf der ersten Seite vor allen anderen Ausgabeobjekten, einschließlich des Seitenkopf-Bereichs, ausgegeben. Auf diese Weise können Sie den Bereich als „Deckblatt“ für den gesamten Report benutzen.

Einen Titelbereich erstellen Sie über die Menüoption **BEREICH | NEUER TITELBEREICH**, und der neue Bereich wird eingerichtet.

Die Überschrift „Inhalt von COMPANY.DBF“ wird als statischer Text betrachtet; einmal in den Report eingebracht, bleibt sein Wert unverändert. Um statischen Text in den Report einzufügen, klicken Sie in der Tastenleiste auf die Taste „Text“ oder aktivieren die Menüoption **OBJEKT | TEXT**. Der Mauszeiger wird zum Texteinfügecursor, mit dem sich die Position des Textausgabeobjekts bestimmen läßt.

Stellen Sie den Mauszeiger in den neuen Titelbereich, und klicken Sie die linke Maustaste, um so die linke obere Ecke des Ausgabeobjekts zu bestimmen. Danach wird die Dialogbox „Text für Textobjekt eingeben“ geöffnet, um Ihren Text aufzunehmen.

Geben Sie „Firmenliste“ bzw. eine andere aussagekräftige Überschrift ein, und klicken Sie auf „OK“, um die Erzeugung des Objekts abzuschließen.

Das neue Ausgabeobjekt sollte in einer für das Auge ansprechenden Weise innerhalb des Report-Entwicklungsbildschirms waagerecht zentriert werden. Das können Sie bewerkstelligen, indem Sie das Textausgabeobjekt manuell mit der Maus in die Bereichsmitte ziehen oder die Menüoption **AUSRICHTEN | ZENTRIERT** ausführen.

Dieser Text erhellt, worum es in dem Report geht, macht aber die Bedeutung der einzelnen Reportspalten nicht klar. Indem Sie die obigen Schritte wiederholen, könnten Sie jedes Feldausgabeobjekt mit einem statischen Textobjekt versehen und so die einzelnen Felder kennzeichnen. Aber dieses Vorgehen würde die End-ausgabe mit unnötigen Wiederholungen nur belasten.

Was dem Report fehlt, sind Spaltenüberschriften – statische Textobjekte mit den Feldnamen, die über die einzelnen Spalten gesetzt werden. Am Anfang einer neuen Seite sollen diese Überschriften im Druck ebenfalls wiederholt werden.

Einen Reportbereich, der oben auf jeder Seite ausgegeben wird, bezeichnen wir als Seitenkopf-Bereich. Jede Seite, mit Ausnahme der ersten (falls ein Titel-bereich angegeben wurde), beginnt unabhängig vom Inhalt der zusammengesetz-ten Datendatei mit einem Seitenkopf-Bereich.

Ein Seitenkopf-Reportbereich läßt sich über die Menüoption **BEREICH | NEUER SEITENKOPF-BEREICH** erzeugen. Dabei wird ein Bereich mit der voreingestellten Höhe von 0,84 cm (0,33") angelegt. Plazieren Sie die folgenden Textausgabe-objekte in diesem Bereich

- ID
- COMPNAME
- CEO

und richten Sie sie manuell über ihren entsprechenden Feldobjekten aus. Wenn Sie damit fertig sind, klicken Sie, um den Einfügemodus zu verlassen, auf „Kein“ in der Tastenleiste (oder betätigen die ESC-Taste).

Der Report-Entwicklungsbildschirm sollte nun in etwa wie in Abb. 2 aussehen; das eingesehene Druckbild sollte weitgehend Abb. 6 unten entsprechen.

Bei einigen Datensätzen wird das letzte Zeichen des Feldes CEO nur zum Teil ausgegeben. Das liegt daran, daß CodeReporter die Größe des Ausgabeobjekts anhand der durchschnittlichen Zeichenbreite der aktuellen Schriftart schätzt. Da diese Schätzung aufgrund von Mittelwerten erfolgt, kann es zu leichten Abwei-chungen kommen, wenn das Feld Großbuchstaben enthält, die in ihrer Breite bekanntlich über dem Durchschnittswert liegen.

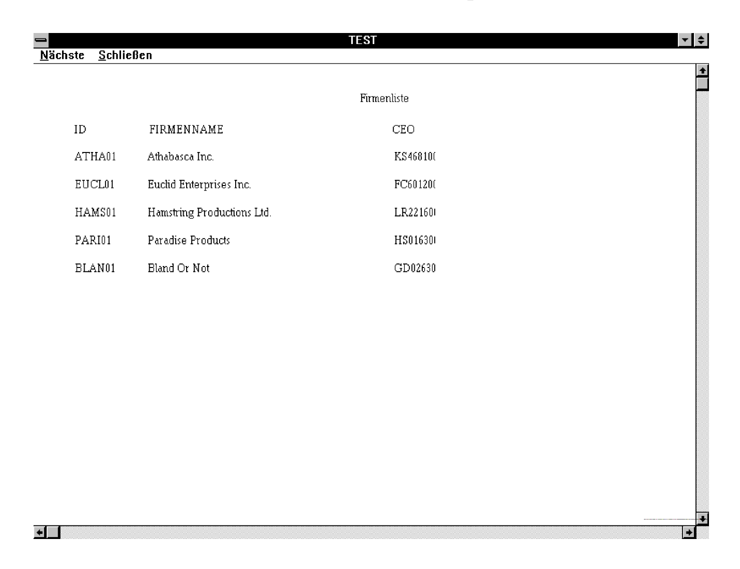


Abb. 6: Fertiggestellter Report von Übung Nr. 2

Um das Feld komplett anzuzeigen, klicken Sie das Feld CEO mit der linken Maustaste an; acht kleine Quadrate erscheinen um das Objekt. Dabei handelt es sich um sogenannte Größenhenkel, mit denen die Größe des Ausgabeobjekts verändert werden kann. Klicken Sie einen der rechten Henkel an, halten die Maustaste gedrückt und ziehen die Maus etwa drei Millimeter nach rechts. Wenn Sie die Maustaste loslassen, wird das CEO-Ausgabeobjekt auf seine neue Breite eingestellt. Bei der Ausgabe wird der Inhalt des Feldes CEO nun vollständig angezeigt.

Kontoauszugsreport

In der folgenden Übung lernen Sie viele der komplexen Eigenschaften von CodeReporter 2.0 bei der Erstellung eines Kontoauszugsreports kennen. Abb. 7 zeigt den vollständigen Report-Entwicklungsbildschirm.

Relation laden

Bei dieser Übung verwenden wir eine Relation, die als Datei TUT3*xxx*.REL gespeichert worden ist. Die Relationsdatei enthält die Verknüpfung der beiden in diesem Report verwendeten Datendateien INVOICES.DBF und CUST.DBF. Beim Laden einer Relationsdatei werden alle offenen Datendateien geschlossen und die in der Relation angegebenen Dateien geöffnet.

Das Laden einer vorhandenen Relation von der Platte bedeutet eine erhebliche Zeitersparnis gegenüber dem manuellen Aufbauen derselben Relation für mehrere Reports.

Laden Sie die Relationsdatei über die Menüoption **DATEI | RELATION LADEN**. Da Relationsdateien indexdateispezifisch angelegt werden, laden Sie die entsprechende Datei, wie in Tabelle 1 angegeben. Wissen Sie nicht, für welche Indexdatei-Kompatibilität CodeReporter installiert wurde, klicken Sie auf die Menüoption **INFO**.

Name der Relationsdatei	Indexkompatibilität
TUT3FOX.REL	FoxPro
TUT3MDX.REL	dBASE IV
TUT3CLI.REL	Clipper
TUT3NDX.REL	dBASE III

Tabelle 1: Kompatibilität der Relationsdateien

Gruppenfelder hinzufügen

Nach dem erfolgreichen Laden der Relation beginnt CodeReporter einen neuen Report, der lediglich die Gruppe „Body“ mit einem einzigen Kopfbereich enthält. In diesem Bereich werden die sich wiederholenden Felder der zusammengesetzten Datendatei ausgegeben.

Versetzen Sie CodeReporter in den Einfügemodus für Feldausgabeobjekte, indem Sie auf „Feld“ in der Tastenleiste klicken bzw. die Menüoption **OBJEKT | FELD** aktivieren. In beiden Fällen erscheint die Listbox „Feldobjekte“, die sämtliche Felder der in der Relation vorhandenen Datendateien anzeigt.

Abb. 7: Fertiggestellter Report von Übung Nr. 4

Markieren Sie folgende Felder der Datendatei INVOICES.DBF innerhalb der Listbox:

- CREDIT
- DEBIT
- ENTERDATE

Bewegen Sie den Mauszeiger in den Bereich der Gruppe „Body“, und klicken Sie die linke Maustaste, um die Feldobjekte zu positionieren. Die Positionierung erfolgt in der Reihenfolge, in der die Felder in der Datendatei vorhanden sind. Klicken Sie in der Listbox „Feldobjekte“ auf „Fertig“, um sie zu schließen.

Um den Anforderungen des Reports gerecht zu werden, müssen Sie die Reihenfolge der Feldausgabeobjekte verändern. Ziehen Sie dazu das Feldobjekt ENTERDATE mit der Maus in die linke obere Ecke des Reportbereichs. Auch die Felder CREDIT und DEBIT lassen sich, wie in Abb. 7 dargestellt, versetzen.

Objektdefinition verändern

Die vielen Standardvoreinstellungen von Ausgabeobjekten müssen nicht in jedem Fall auch zutreffen, wie das bei den Feldern ENTERDATE, CREDIT und DEBIT der Fall ist. ENTERDATE wird im Datumsformat MM/DD/YY ausgegeben, während es als DD MMM CCYY erscheinen sollte, und sowohl CREDIT als auch DEBIT sollen nicht ausgegeben werden, wenn sie gleich null sind.

Markieren Sie das Ausgabeobjekt ENTERDATE, und betätigen Sie die Eingabetaste, um das Objektmenü aufzurufen. In diesem Menü wählen Sie die Option **OBJEKTDEFINITION** an, die die Dialogbox „Objektdefinition“ aufruft, innerhalb derer das Datumsformat und die Größe des Objekts verändert werden können.

Öffnen Sie die einzeilige Listbox „Datumsformat“, und wählen Sie darin das Format DD MMM CCJJ an (oder geben Sie es per Hand ein). Setzen Sie die „Breite“ (im Teilbereich „Größe“) auf 1,9 cm. Wenn Sie nun auf „OK“ klicken, werden diese beiden Veränderungen vorgenommen, und das Objekt wird entsprechend angepaßt.

Aktivieren Sie, wie oben beschrieben, die Dialogbox für das Ausgabeobjekt „CREDIT“. Die Optionsschaltfläche „Null anzeigen“ im rechten unteren Bereich der Dialogbox ist ausgewählt. Klicken Sie auf die Schaltfläche, um sie zu deaktivieren, und auf „OK“, um die Dialogbox zu schließen. Wiederholen Sie diese Schritte auch für das Ausgabeobjekt DEBIT.

Größe von „Body“ verändern

Lassen Sie sich mit der Menüoption **DATEI | DRUCKBILD EINSEHEN** den Report anzeigen. Zwischen den ausgegebenen Zeilen ist jeweils eine Leerzeile vorhanden. Das liegt daran, daß die Gruppe mit der voreingestellten Höhe von 0,84 cm (0,33") angelegt wurde, die Buchstaben für das Ausgabeobjekt aber lediglich etwa 0,43 cm (0,17") hoch sind. Um die Leerzeilen zu entfernen, kann man die Höhe des Reportbereichs an die des Ausgabeobjekts angleichen.

Die Angleichung kann entweder über die Größenhenkel des Bereichs oder die unmittelbare Eingabe der Größe in der Dialogbox „Bereich bearbeiten“ erfolgen. Hier erläutern wir die erste Methode, während die zweite weiter unten bei der Bearbeitung des Gruppenkopfbereichs „Kunde“ eingesetzt wird.

Ziehen Sie den linken bzw. rechten Henkel des Bereichs „Body“ mit der Maus, bis der Bereich kleiner ist als der der drei Ausgabeobjekte. Wenn Sie die Maustaste loslassen, versucht CodeReporter, den Bereich kleiner als die Reportobjekte zu machen, aber gibt die in Abb. 8 angezeigte Warnung aus.

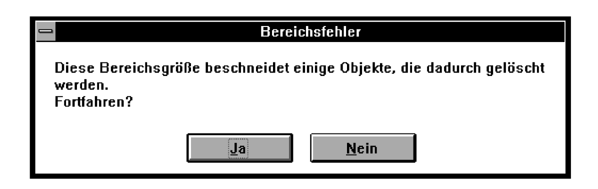


Abb. 8: Bereichsfehlermeldung

Klicken Sie auf „Nein“. Auf diese Weise wird CodeReporter veranlaßt, den Bereich auf die kleinstmögliche Größe einzustellen, ohne daß dabei ein Ausgabeobjekt beschnitten (und gelöscht) wird. (Im Kapitel „Bereiche“ finden Sie weitere Einzelheiten zur Größe von Bereichen.)

Lassen Sie sich den Report (**DATEI | DRUCKBILD EINSEHEN**) noch einmal anzeigen: die Leerzeilen sind verschwunden.

Die Gruppe „Kunde“ hinzufügen

So wie der Report jetzt aussieht, macht er für einen unbeteiligten Leser nicht viel Sinn, da lediglich einige Daten und Zahlenwerte aufgeführt werden. Ohne Kenntnis darüber, auf wen sich diese Zahlen beziehen und was sie bedeuten, vermittelt der Report keine echten Informationen.

Die möglicherweise wichtigste Information, die dem Report noch fehlt, besteht in der Zuordnung der Soll- und Habenwerte zu Personen. Das geschieht über das Anlegen einer zweiten Gruppe mit der Menüoption **GRUPPE | NEU**; in den Bereich dieser Gruppe werden dann die den einzelnen Kunden betreffenden Informationen gesetzt.

Nach dem Erzeugen der neuen Gruppe erscheint die Dialogbox „Gruppendefinition“ (Abb. 9), um dem Reportentwickler Gelegenheit zu geben, einige der Gruppenvoreinstellungen zu abzuändern.

CodeReporter verleiht den Gruppen in einem Report automatisch einen eindeutigen Namen. Logischerweise kann diese Bezeichnung den Zweck der Gruppe nicht treffend beschreiben. In der Dialogbox „Gruppendefinition“ können Sie den Namen der Gruppe im Editierfeld „Gruppenname“ verändern. Geben Sie anstelle der voreingestellten Bezeichnung „Gruppe 2“ „Kunde“ ein.

Da diese Information nicht für jeden einzelnen Soll- und Habenwert angezeigt werden muß, wird ein Gruppenrücksetzausdruck definiert, um den Bereich der Gruppe Kunde nur dann auszugeben, wenn sich der Kunde ändert. Ein Gruppenrücksetzausdruck, der die Untermenge, zu der ein einzelner Kunde gehört, eindeutig identifiziert, lautet zum Beispiel:

INVOICES->CUSTID

Abb. 9:
Dialogbox „Gruppendefinition“ für die Gruppe Kunde

Diesen Ausdruck schreiben Sie in das Eingabefenster „Rücksetzbedingung“.

Immer wenn sich die Identifizierungsnummer des Kunden ändert, sollen die Soll- und Habenwerte einem anderen Kunden zugeordnet werden. Diese auch als Gruppenrücksetzbedingung bezeichnete Bedingung bewirkt die Ausgabe der Gruppe Kunde.

Für die voreingestellte Gruppe „Body“ wurde keine Gruppenrücksetzbedingung definiert. Das liegt daran, daß die Gruppe „Body“ ohne Gruppenrücksetzausdruck für jeden zusammengesetzten Datensatz zurückgesetzt wird; dabei wird der Gruppenkopf-Bereich für jeden einzelnen Datensatz ausgegeben.

Aktivieren Sie innerhalb der vorliegenden Dialogbox ebenfalls die Schaltflächen „Kopf austauschen“, „Fuß austauschen“ und „Seitenzahl zurücksetzen“. Diese Optionen stellen dem Kontoauszugsreport einige besondere Bearbeitungsmöglichkeiten zur Verfügung. Nähere Einzelheiten zu diesen Einstellungen entnehmen Sie bitte dem Kapitel „Gruppen“.

Klicken Sie auf „OK“, um die Dialogbox „Gruppendefinition“ zu schließen.

Kopf der Gruppe „Kunde“ füllen

Der Kunden-Kopfbereich wird aufgrund der aktivierten Option „Kopf austauschen“ oben auf jeder Seite ausgegeben, die mit einem neuen Kunden beginnt. Aus diesem Grund ist es erforderlich, sowohl in den Report selbst als auch den Kopfbereich der Gruppe einen Erläuterungstext einzufügen.

Erstellen Sie ein Textausgabeobjekt mit dem Inhalt „KONTOAUSZUG“, setzen Sie es in den Kunden-Kopfbereich, und richten Sie den Text mit der Menüoption **AUSRICHTEN | ZENTRIERT** waagerecht mittig aus.

Ehe der Kunden-Kopfbereich mit den restlichen Texten versehen wird, muß die Bereichsgröße verändert werden. Das können Sie entweder mit der Maus über die Größenhenkel, wie weiter oben beschrieben, oder über die Dialogbox „Bereich bearbeiten“ bewerkstelligen.

Die Dialogbox wird aufgerufen, wenn Sie die rechte Maustaste auf einer leeren Stelle des Kunden-Bereichs (nicht im Gruppeninformationsfenster) klicken.

Geben Sie in das Editierfeld „Höhe“ (siehe Abb. 10) den Wert 3,3 cm ein, und klicken Sie auf „OK“. Die Größe des Kopfbereichs wird automatisch neu eingestellt.

Die Felder mit den Kundendaten werden über die Listbox „Feldobjekte“ in den neuen Bereich eingefügt. Die Listbox rufen Sie mit der Taste „Feld“ in der Tastenleiste auf. Verschieben Sie die Bildlaufleiste, bis folgende Felder zu sehen sind:

NAME – der Name des Kunden
ADDRESS – die Anschrift des Kunden
CITYSTZIP – Ort, Staat und Postleitzahl des Kunden

Diese Felder können jeweils einzeln markiert und an der entsprechenden Stelle im Report abgelegt (siehe Abb. 7) oder als Gruppe markiert und zusammen abgelegt werden.

Abb. 10: Dialogbox „Bereich bearbeiten“

Markieren Sie die obigen Felder als Ganzes, setzen Sie den Mauszeiger in den Kunden-Kopfbereich, und klicken Sie die linke Maustaste. Jetzt wird die Dialogbox „Feldgestaltung“ geöffnet. Klicken Sie nicht wie bei der Gruppe „Body“ einfach auf „OK“, sondern markieren Sie das Kontrollkästchen „Senkrecht“. Wenn Sie nun auf „OK“ klicken, werden die drei markierten Felder automatisch linksbündig untereinander ausgerichtet.

Plazieren Sie des weiteren zwei Textobjekte mit dem Inhalt „Soll“ bzw. „Haben“ in den unteren Teil des Kunden-Kopfbereichs über die Feldobjekte DEBIT und CREDIT.

Der Seitenkopf-Bereich

Erstellen Sie nun mit Hilfe der Menüoption **BEREICH | NEUER SEITENKOPF-BEREICH** einen Seitenkopf, der auf jeder Seite des Reports oben ausgegeben wird (bis auf die Seiten, auf denen er gegen den Kunden-Gruppenkopf ausgetauscht wird). Dieser Bereich hat die Aufgabe, den Report kurz zu beschreiben und alle Seiten miteinander zu verbinden.

Stellen Sie die Bereichshöhe auf 1,3 cm ein. Der Bereich enthält vier Ausgabeobjekte, und zwar ein Feld-, zwei Text- und ein Ausdrucksobjekt. In der Listbox „Feldobjekte“ (die noch zu sehen sein sollte) markieren Sie das Feld NAME und fügen es in den Seitenkopf-Bereich ein. Schließen Sie die Listbox, indem Sie auf die Schaltfläche „Fertig“ klicken.

Die beiden für den Kunden-Kopfbereich erstellten Textobjekte „Soll“ und „Haben“ können wie beim Kundenkopf manuell erzeugt und in den Seitenkopf-Bereich eingefügt werden.

Alternativ dazu können Sie die Objekte im Kundenkopf aber auch markieren und dann über die Menüoption **BEARBEITEN | KOPIEREN** eine Kopie davon in der Zwischenablage ablegen. Wählen Sie darauf die Option **BEARBEITEN | EINFÜGEN** an, stellen den Mauszeiger in den Seitenkopf-Bereich und klicken die linke Maustaste, um die Textobjekte zu positionieren.

Das vierte Ausgabeobjekt, ein Ausdruck, dient der Ausgabe der Reportseitenzahl. Klicken Sie auf die Schaltfläche „Ausdruck“ auf der Tastenleiste (CodeReporter stellt sich auf den Einfügemodus für Ausdrucksausgabeobjekte ein), stellen Sie den Mauszeiger rechts oben in den Seitenkopf-Bereich, und drücken Sie die linke Maustaste. Dadurch wird die Dialogbox „Ausdruck eingeben“ aktiviert, in die der Ausdruck für das Ausdrucksausgabeobjekt eingegeben werden kann.

Bitte geben Sie den folgenden Ausdruck (einschließlich der Anführungszeichen) ein:

```
"Seite: "+STR(PAGENO(), 3,0)
```

Dieser Ausdruck, der sich aus einer Zeichenkette (`"Seite:"`) und der Seitenzahl (`"PAGENO()"`) zusammensetzt, könnte auch aus zwei Objekten gebildet werden, einem Textobjekt für „`Seite:`“ und einem separaten Ausdrucksausgabeobjekt für „`PAGENO()`“. Da beide Objekte jedoch logisch zusammengehören und immer gemeinsam versetzt werden, ist es nur sinnvoll, sie in einem Ausdruck unterzubringen.

Klicken Sie nun auf „OK“, um die Positionierung des Ausgabeobjekts abzuschließen.

Der Schlußbereich des Kontoauszugs

Der vorliegende Kontoauszug führt die Summe der Soll- und Habenwerte für das Konto jedes einzelnen Kunden auf. Um den Kunden den Überblick zu erleichtern, enthält der Auszug eine Zeile, aus der hervorgeht, ob sie der Firma Geld schulden bzw. zuviel bezahlt haben.

Da ein Kunde sich nicht gleichzeitig im Zahlungsrückstand und auf der Guthabenseite befinden kann, müssen diese Werte in Reportbereiche geschrieben werden, die sich gegenseitig ausschließen.

Beim Anlegen der Kunden-Gruppe hat CodeReporter automatisch ebenfalls einen einzigen Gruppenfuß-Bereich angelegt. Dieser Bereich wird für die Ausgabe der summierten Soll-/Habenwerte des Kunden verwendet. Es besteht die Möglichkeit, solange die Kunden-Gruppe markiert ist, über die Menüoption **BEREICH | NEUER FUSSBEREICH** zwei weitere Fußbereiche („Sie schulden uns“ und „Sie haben gut“) anzulegen.

Erweitern Sie jeden Kundengruppen-Fußbereich um ein Textobjekt (s. Abb. 7):

Summe Soll/Haben
Sie haben gut
Sie schulden uns

Um zu bestimmen, wer wem Geld schuldet, muß jeweils die Gesamtsumme der Soll- und Habenwerte des Kundenkontos ermittelt und miteinander verglichen werden.

Summen werden mit Hilfe der Listbox „Summenberechnungen“ gebildet, die über die Schaltfläche „Summe“ auf der Tastenleiste aktiviert wird. Die Listbox enthält neben den Namen aller numerischen Felder der zusammengesetzten Datendatei auch alle numerischen Berechnungen. Ein Ausgabeobjekt für das Feld CREDIT positionieren Sie, indem Sie den Eintrag CREDIT in der Listbox markieren und diesen in den ersten Fußbereich der Kunden-Gruppe einfügen. Das ruft die Dialogbox „Summe bilden“ auf den Plan (Abb. 11), die die voreingestellten Werte der Summe enthält.

Der Dialog „Summe bilden“ enthält alle notwendigen Einstellungen (Rücksetzausdruck, Art der Summe), um die Summe des Feldes CREDIT zu bilden. Die Bezeichnung **SUMME0** jedoch beschreibt nur unvollkommen, was zusammengerechnet wird. Ändern Sie also die Bezeichnung **SUMME0** im Editierfeld „Summenname“ in **CREDIT_SUM** ab, und klicken Sie auf „OK“.



Abb. 11: Dialogbox „Summe bilden"

Wiederholen Sie die obigen Schritte, um für das Feld DEBIT im ersten Fußbereich der Kunden-Gruppe ein Summenausgabeobjekt einzufügen, und bezeichnen Sie diese Summe als **DEBIT_SUM**.

Der geschuldete Geldbetrag läßt sich über ein Berechnungsausgabeobjekt ermitteln (und ausgeben), das die Differenz zwischen den Summen **CREDIT_SUM** und **DEBIT_SUM** errechnet. Diesen Rechenvorgang können (und werden) wir auch in anderen dBASE-Ausdrücken verwenden.

Ein Berechnungsausgabeobjekt wird mit Hilfe der Listbox „Berechnungsobjekt" gebildet, das über die Schaltfläche „Berechnung" auf der Tastenleiste aktiviert wird. Klicken Sie auf den Menüpunkt „Neue Ber.", um die Dialogbox „Berechnung" aufzurufen (Abb. 12).

Der Name der Berechnung wird zur Identifizierung der Berechnung in weiteren dBASE-Ausdrücken verwendet. Schreiben Sie „SOLL_HABEN" in das Editierfeld „Name der Berechnung" sowie den folgenden Ausdruck in das Editierfeld „Berechnungsausdruck":

```
CREDIT_SUM() - DEBIT_SUM()
```

Klicken Sie auf „OK", um das Editierfeld zu schließen und in die Dialogbox „Berechnungsobjekt" zurückzukehren. Markieren Sie in der Listbox die neue

Berechnung **SOLL_HABEN**, und positionieren Sie sie im zweiten Fußbereich der Kunden-Gruppe („Sie haben gut").



Abb. 12: Dialogbox „Berechnungsobjekt"

Schuldet der Kunde der Firma Geld, enthält das Ergebnis der Berechnung **SOLL_HABEN** eine Minuszahl. Um den dritten Reportbereich („Sie schulden uns") jedoch richtig auszugeben, sollte der Wert eine positive Zahl sein. Erzeugen Sie im dritten Fußbereich der Kunden-Gruppe also ein Ausdrucksausgabeobjekt (wie oben die Seitenzahl), und geben Sie folgenden Ausdruck ein:

```
SOLL_HABEN() * -1
```

Dieser Ausdruck ergibt in allen Fällen, in denen ein Kunde der Firma Geld schuldet, eine positive Zahl.

Fußbereiche unterdrücken

Zum gegenwärtigen Zeitpunkt gibt der Report je einmal alle Soll- und Habenwerte für einen Kunden, die Summenwerte für Soll und Haben und eine Zeile aus, in der steht, daß die Firma dem Kunden Geld schuldet und umgekehrt; jedoch nur eine dieser Aussagen kann stimmen. Das heißt, daß die beiden letzten Bereiche bedingt unterdrückt werden müssen, wobei die Bedingung vom Wert der Berechnung **SOLL_HABEN()** abhängt.

Klicken Sie die rechte Maustaste auf einer leeren Stelle im zweiten Fußbereich der Kunden-Gruppe („Sie haben gut") an, wird die Dialogbox „Bereich bearbeiten" aufgerufen. Schreiben Sie den folgenden Ausdruck in das Editierfeld „Unterdrückungsbedingung", und klicken Sie auf „OK":

```
SOLL_HABEN() <= 0
```

Auf diese Weise wird dem Report angezeigt, daß der zweite Bereich („Sie haben gut“) nicht ausgegeben werden soll, wenn die Guthabensumme minus Schuldbetrag kleiner oder gleich null ist (der Kunde schuldet der Firma Geld). Verfügt der Kunde über ein Guthaben, gibt die Berechnung **SOLL_HABEN** einen positiven Wert zurück, so daß der Reportbereich ausgegeben, d. h. nicht unterdrückt, wird.

Für den dritten Bereich sollte folgende Unterdrückungsbedingung verwendet werden:

```
SOLL_HABEN() > 0
```

Die letzte Aufgabe in diesem Übungsreport besteht in der Bearbeitung des Zahlenformats der numerischen Ausgabeobjekte der Fußbereiche (zwei Summen, eine Berechnung und ein Ausdruck). Rufen Sie das jeweilige Objektmenü auf (Objekt mit rechter Maustaste anklicken bzw. Eingabetaste betätigen, wenn Objekt markiert ist), und wählen Sie die Menüoption **OBJEKTDEFINITION** an. Bearbeiten Sie folgende Einstellungen:

1. Das Editierfeld „Nachkommastellen“. Voreingestellt ist „0“. Geben Sie statt dessen „2“ ein.
2. Die Optionsschaltfläche „Zahlenformat“. Das Ausgabeobjekt soll im Format „Währung“ erscheinen.

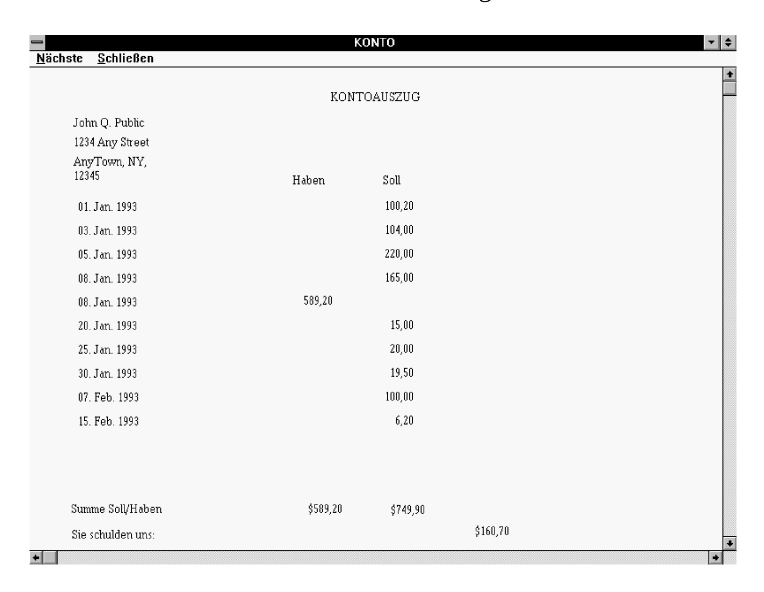
Den Report einsehen

Über die Menüoption **DATEI | DRUCKBILD EINSEHEN** haben Sie die Möglichkeit, sich die Seiten des Reports auf dem Bildschirm anzeigen zu lassen. Die ersten drei sollten in etwa so aussehen, wie in Abb. 13 angegeben.

Einen Sortierausdruck angeben

Wie aus den ersten drei Seiten des Reports (Abb. 13) hervorgeht, teilt der Kontoauszug von Customer O'Mine den von John Q. Public in zwei Teile. Das ist der Fall, weil CodeReporter die Datensätze in ihrer natürlichen Reihenfolge (das heißt, wie sie physisch vorhanden sind) aus der zusammengesetzten Datendatei abruft.

Der letzte Schritt beim Anlegen dieses Reports besteht darin, die Datensätze der zusammengesetzten Datendatei nach der Kundennummer zu ordnen, die im Feld CUSTID zu finden ist.



KONTO

Nächste Schließen

KONTOAUSZUG

John Q. Public
1234 Any Street
AnyTown, NY,
12345

	Haben	Soll	
01. Jan. 1993		100,20	
03. Jan. 1993		104,00	
05. Jan. 1993		220,00	
08. Jan. 1993		165,00	
08. Jan. 1993	589,20		
20. Jan. 1993		15,00	
25. Jan. 1993		20,00	
30. Jan. 1993		19,50	
07. Feb. 1993		100,00	
15. Feb. 1993		6,20	
Summe Soll/Haben	$589,20	$749,90	
Sie schulden uns:			$160,70

Abb. 13, Seite 1



KONTO

Nächste Schließen

KONTOAUSZUG

Customer O'Mine
54321 MultiWealth Way
Amberville, CA,
90122

	Haben	Soll	
22. Feb. 1993		160,00	
Summe Soll/Haben	$,00	$160,00	
Sie schulden uns:			$160,00

Abb. 13, Seite 2

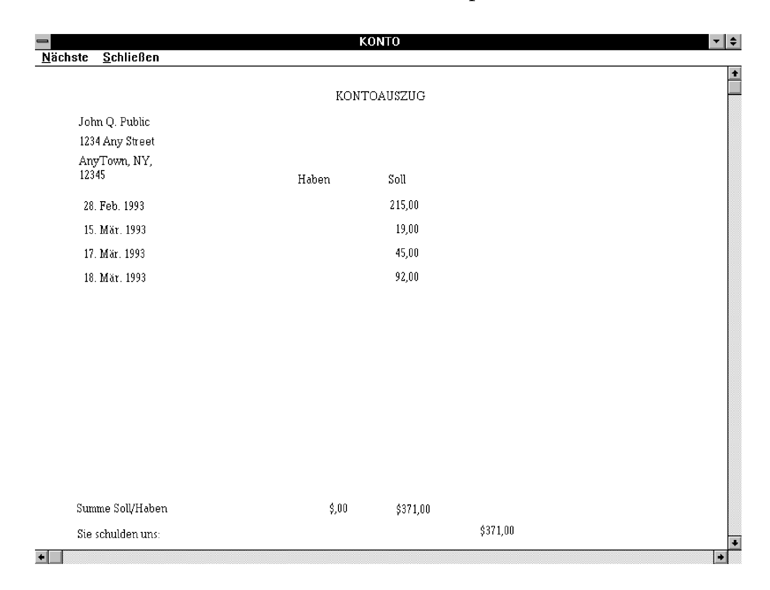


Abb. 13, Seite 3

Rufen Sie über die Menüoption **REPORT | SORTIERAUSDRUCK** die Dialogbox „Ausdruck eingeben“ zur Eingabe des Sortierausdrucks auf. Geben Sie folgenden Ausdruck ein, und klicken Sie auf „OK“:

```
INVOICES->CUSTID
```

Wenn Sie sich den Report das nächste Mal anzeigen lassen, wird der neue Sortierausdruck berücksichtigt, und alle Soll- und Habenwerte von John Q. Public werden vor denen von Customer O'Mine angezeigt.

1 Einen Report entwerfen

Die Erstellung eines Reports erfolgt in drei Schritten. Zunächst muß man sich darüber im klaren sein, was der Report enthalten und wie er aussehen soll. Die Schritte zwei und drei bestehen darin, ihn zu entwickeln und probehalber auszugeben. Stimmt das Endprodukt nach dem dritten Schritt mit der Ausgangsidee überein, ist die Entwicklung abgeschlossen. In den meisten Fällen allerdings wird der Durchführungsschritt mehrere Male wiederholt, bis der Report in seiner endgültigen Form vorliegt.

Noch öfter aber kommt es vor, daß sich die Vorstellung über Inhalt und Aufbau im Laufe der Entwicklungsarbeit grundlegend verändert. Die Ursache dafür ist eine unzureichende Planung vor der eigentlichen Reportdurchführung.

Das vorliegende Kapitel befaßt sich mit den Überlegungen, die der Entwicklung eines Reports vorangestellt werden müssen.

Zweck des Reports

Wenn Sie einen Report entwickeln, müssen Sie genau wissen, warum er erstellt wird. Ohne eine klare Vorstellung darüber zu haben, warum der Report überhaupt erforderlich ist, ist es relativ einfach, einen ansehnlichen, neuen, klaren – völlig nutzlosen – Report vorzulegen.

Wenn die Lagerverwaltung z. B. wissen will, wie viele XYZs sie nachbestellen muß und ihr ein Report mit dem XYZ-Verbrauch vom letzten Jahr vorgelegt wird, wird sie geringfügig aus der Fassung geraten, und der Report muß überarbeitet werden.

Bei der Zweckerklärung handelt es sich um eine kurze Beschreibung der Erfordernisse eines Reports. In ein oder zwei Zeilen wird dabei kurz angegeben,

- wer den Report angefordert hat,
- welche Informationen er enthalten soll,
- für welche Eckdaten er gelten soll (z. B. Zeiträume, Produkte, Örtlichkeiten usw.),
- wie er eventuell zu formatieren ist.

Diese Erfordernisse können sich einfach, z. B. *„Die Personalabteilung benötigt eine Liste aller Personen in der Datendatei EMPLOYEE“*, oder auch komplex gestalten, z. B. *„Der Aufsichtsrat benötigt für jede einzelne Niederlassung jeder Firma innerhalb des Konzerns eine Übersicht über Erträge und Aufwendungen der letzten zwei Jahre – alphabetisch nach Firma und Geschäftsbezeichnung sortiert.“*

Die Erstellung einer solchen Zweckerklärung hilft dabei, die Anforderungen an den Report zu erhellen; später kann man anhand der Erklärung überprüfen, ob aus dem Report tatsächlich das hervorgeht, wozu er entwickelt worden ist.

Entwurf auf dem Papier

Nach der Veränderung des Inhalts ist das Erscheinungsbild eines Reports der Aspekt, der am häufigsten Wandlungen unterworfen ist. Je eher Sie sich für ein definitives Layout entscheiden, desto weniger Zweifel an Ihrer Arbeit werden sich bei Ihnen einstellen; darüber hinaus geht die eigentliche Erstellung des Reports erheblich schneller vonstatten.

Die einfachste Art und Weise, das Format eines Reports zu ermitteln, besteht darin, ihn rasch auf einem Blatt Papier zu skizzieren und dabei Musterdaten zu verwenden. Bei dem Entwurf ist die Anordnung der Daten wichtiger als deren Genauigkeit.

Die Stelle, an die eine Information gesetzt wird, ist oft eine Kombination aus Firmenpolitik und persönlichem Geschmack – also gibt es keine Regeln. Bei einigen Firmen stehen die Seitenzahlen oben, bei anderen unten. Fertigen Sie verschiedene Entwürfe an, bis ein geeigneter dabei ist.

Der wichtigste Gesichtspunkt eines Entwurfs ist, daß er dem Anspruch der Zweckerklärung volles Genüge leistet. Ist das nicht der Fall, muß der Entwurf überarbeitet werden.

Abb. 1.1 zeigt einen Reportentwurf für folgende Zweckerklärung: *„Die Fördergesellschaft benötigt einen Report, der die Spendensumme des laufenden Jahres und eine alphabetische Aufstellung aller Spender mit Anschrift, Spendernummer und Betrag enthält, die in diesem Jahr $ 1.000,- oder mehr gespendet haben.“*

Förderergesellschaft Spenderliste 1. Januar 1993 bis 21. Mai 1993		
Gesamtspenden der Förderer: $ xxxxxxx		
Spender	Spender-Nr.	Summe
Adams, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 1.000
Baker, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 9.000
Cramford, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 6.000
Denver, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 6.000
Evans, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 3.000
Finnigan, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 15.000
Goodbody, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 2.000
Hamilton, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 1.000
Il, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 5.000
Jorganson, John 123 West 4th Street, Ort, ST, 55212	4321-34-1234	$ 7.000

Abb. 1.1: Reportentwurf

Der Reportentwurf in Abb. 1.1 leistet der Zweckerklärung voll Genüge. Aufgeführt werden oben der Urheber, die Spendengesamtsumme und Name, Anschrift und Spender-Nr. in alphabetischer Reihenfolge.

Erst wenn dieser Reportentwurf vom Urheber, in diesem Fall der Förderergesellschaft freigegeben wird, kann mit dem eigentlichen Aufbau des Reports begonnen werden.

Den Entwurf auswerten

Der fertige Entwurf enthält sämtliche für den Report erforderliche Informationen. Sicherlich ist an dieser Stelle die Versuchung groß, CodeReporter aufzurufen und sofort mit der Arbeit am Report zu beginnen. Es sind jedoch weitere Planungsschritte erforderlich, um den Einsatz von CodeReporter hinreichend produktiv zu gestalten.

Zunächst geht es darum, den Entwurf auszuwerten und die einzelnen Elemente typenmäßig festzulegen. Es sind Fragen zu bedenken, wie: Ändert sich diese Information innerhalb des Reports? Besteht die Möglichkeit, daß diese Information bei jeder Ausgabe des Reports anders lautet? Wird diese Summe berechnet? Ermitteln Sie anhand des Reportentwurf die einzelnen Elemente, und bezeichnen Sie sie mit Ausdrücken (siehe Abb. 1.2) wie z. B.:

- Db-Feld. Diese Daten stammen aus einer Datenbank.
- Grafik bezeichnet ein als Datei abgelegtes Bild. Verzichten Sie auf Elemente (Briefkopf usw.), die Ihr Drucker nicht mit ausgibt.
- Statischer Text. Dieser Text ändert sich innerhalb des Reports nicht.
- Berechneter Text/Summe. Diese Informationen setzen sich aus zwei oder mehr Elementen des Reports zusammen.
- Summe. Diese Information summiert numerische Daten auf.
- Linien.
- Sonstige. Elemente wie Seitenzahlen oder aktuelles Datum/aktuelle Uhrzeit, die in keine andere Kategorie passen.

Wenn der Report dann tatsächlich mit CodeReporter in Angriff genommen wird, beschleunigen diese Definitionen die Positionierung der einzelnen Elemente.

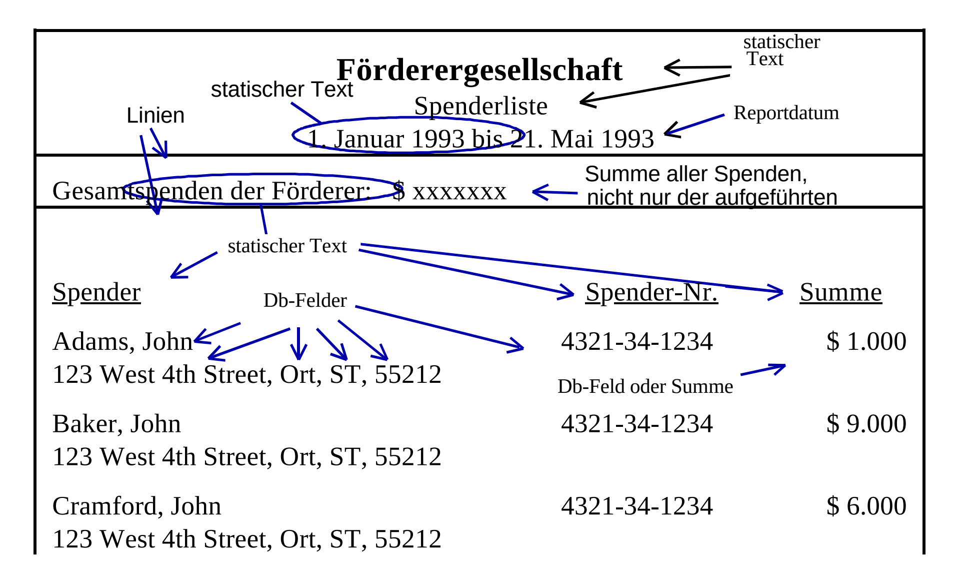


Abb. 1.2: Bearbeiteter Reportentwurf

Zusammengehörige Reportbereiche ermitteln

Kreisen Sie auf einer Kopie des Reportentwurfs die Bereiche ein, die definitiv zusammengehören. Ein solcher Abschnitt könnte z. B. lediglich am Anfang bzw. Ende gedruckt werden, auf jeder Seite wiederholt werden, hier und da im Report auftauchen oder kontinuierlich vorhanden sein.

Diese Abschnitte, sogenannte Bereiche, legen fest, wann und warum zusammengehörige Teile des Reports gemeinsam ausgegeben werden. Je eher man diese Bereiche definiert, desto klarer tritt der Aufbau des Reports zu Tage – und desto schneller steht der endgültige Aufbau des Reports fest.

Nachdem Sie die Abschnitte markiert haben, schreiben Sie in Kurzform daneben, wann der entsprechende Bereich ausgegeben werden soll. Aus Abb. 1.3 gehen die für unseren Musterreport erforderlichen Bereiche hervor; daneben wird kurz erläutert, warum sie so definiert worden sind.

Förderergesellschaft
Spenderliste
1. Januar 1993 bis 21. Mai 1993

Gesamtspenden der Förderer: $ xxxxxxx

Dieser Bereich wird nur am Anfang des Reports ausgegeben.

Spender	Spender-Nr.	Summe
Adams, John 123 West 4th Street, Ort, ST, 55212	4321-34- 1234	$ 1.000
Baker, John 123 West 4th Street, Ort, ST, 55212	4321-34- 1234	$ 9.000
Cramford, John 123 West 4th Street, Ort, ST, 55212	4321-34- 1234	$ 6.000
Denver, John 123 West 4th Street, Ort, ST, 55212	4321-34- 1234	$ 6.000

Dieser Bereich wird auf jeder Seite ausgegeben. Nur auf der 1. Seite steht er nach dem Titel.

Dieser Bereich wird für jeden Spender einmal ausgegeben.

Abb. 1.3: Reportentwurf mit Gruppen

Die Elemente des Reports festlegen

Der Entwurf eines Reports ist auf dem Weg bis zu seiner Erstellung die einfachste Aufgabe. Schwieriger ist es da schon, die Daten zur Verfügung zu haben, die in den Report einfließen sollen. Dazu sind Kenntnisse (oder eine Liste) aller Datendateien erforderlich, die möglicherweise reportrelevante Informationen enthalten.

Die statischen Textelemente können wir vernachlässigen, da sie bei der Entwicklung des Reports unmittelbar in den CodeReporter eingegeben werden. Linien, Rahmen und Zusatzelemente können wir zu diesem Zeitpunkt ebenfalls vernachlässigen, da auch sie mit CodeReporter erstellt werden. Übrigbleiben die Elemente, die aus Datendateien stammen, sowie Grafiken.

Wenn Sie eine Liste der besonderen Reportelemente zusammenstellen, die auch die Dateien mit eben diesen Elementen enthält, wissen Sie sogleich, welche Daten- und Grafikdateien für den Report erforderlich sind. Auf diese Weise stellen

Sie sicher, daß die notwendigen Dateien vorhanden sind und schließen nicht erforderliche von vornherein aus.

Für Abb. 1.1 lautet die Liste der Reportelemente, die Daten aus einer Datendatei verwenden:

- $ xxxxxxx – Gesamtsumme der Spenden der Förderer
- John Q. Public – der Name des Spenders
- 123 West 4th Street, Ort, ST, 55212 – Spenderanschrift, die sich aus Straße, Ort, Staat und Postleitzahl zusammensetzt
- 4321-34-1234 – Spender-Nr. und
- $ ******* – Gesamtsumme der Spenden eines Spenders

Je nachdem, wie die Datenbanken aufgebaut sind, kann sich die Gestaltung des Reports als einfach bzw. kompliziert erweisen.

Wenn die Datendatei so organisiert ist wie SPENDER.DBF (siehe Abb. 1.4), dann befinden sich alle für den Report erforderlichen Datenfelder in dieser einen Datei, so daß der Report im Grunde genommen eine Auflistung der in dieser vorhandenen Daten ist.

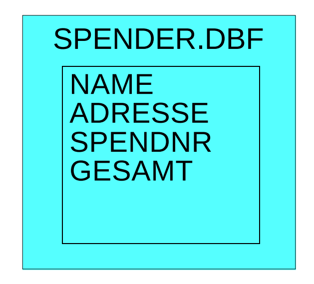


Abb. 1.4

Leider sind die Daten für einen Report im allgemeinen nicht so gut vorbereitet, sondern meistens über mehrere verknüpfte Datendateien verteilt. Selbst in unserem einfachen Beispiel könnten die erforderlichen Datendateien aussehen, wie in Abb. 1.5 angegeben.

Da die Daten in den meisten Fällen aus mehreren Datendateien stammen, verwenden wir für den Rest des vorliegenden Kapitels die Abb. 1.5 als Muster.

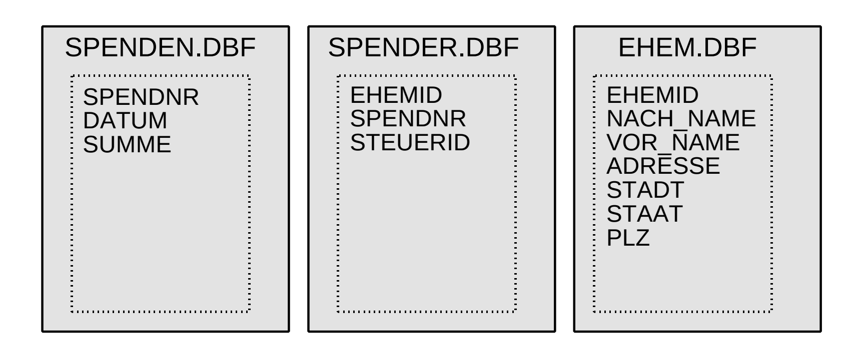


Abb. 1.5

Als nächster Arbeitsschritt werden die Reportelemente, die von Datendateien abhängen, den entsprechenden Datendateien zugeordnet. Tabelle 1.1 enthält eine solche Übersicht. Einige Elemente, z. B. Spendername, setzen sich aus mehreren Feldern zusammen, während andere, z. B. die Spender-Nr., in mehreren Datendateien vorkommen. Sie sollten alle Elemente aufführen, da sie beim Aufbau des Reports alle erforderlich werden könnten.

Element	Datenfelder für Element
$ xxxxxxx	SPENDEN->GESAMT
Adams, John	EHEM>NACH_NAME, EHEM->VOR_NAME
123 West 4th Street, AnyTown, ST, 55212	EHEM->ADRESSE, EHEM->STADT, EHEM->STAAT, EHEM->PLZ
4321-34-1234	SPENDER->SPENDNR, SPENDEN->SPENDNR
$ *******	SPENDEN->SUMME

Tabelle 1.1

Die Relation planen

Die aufgeführten Datenfelder sind für den Report erforderlich. Befinden sich alle in derselben Datendatei, können Sie diesen Arbeitsschritt überspringen und die Reporterstellung mit CodeReporter unmittelbar in Angriff nehmen.

Wie bereits gesagt, greifen die meisten Report auf Daten aus mehreren Datendateien zurück. Die Verknüpfung, bzw. Verkettung, dieser Datendateien ist der eigentliche Kern von relationalen Datenbanken. Das Modell relationaler Datenbanken und wie man mit CodeReporter eine Relation erstellt, behandeln wir ausführlich in Kapitel 2.

Verbinden Sie in der Übersicht über die Datendateien und ihre Felder die gemeinsamen Felder mit Linien. Daraus werden die erforderlichen Verbindungen zwischen den Datendateien ersichtlich; darüber hinaus können die zwischen den Datendateien bestehenden Relationen leichter bestimmt und die Top-Master-Datendatei des Reports ermittelt werden.

Beachten Sie die in Abb. 1.6 von links nach rechts verlaufende Relation. Die Datendatei SPENDEN.DBF enthält (oder könnte enthalten) für jeden Spender mehr als einen Spenden-Datensatz. In der Datendatei SPENDER.DBF hat jeder Ehemalige lediglich einen Datensatz, das heißt, daß alle Spenden eines Ehemaligen unter einer einzigen Kennummer zu finden sind.

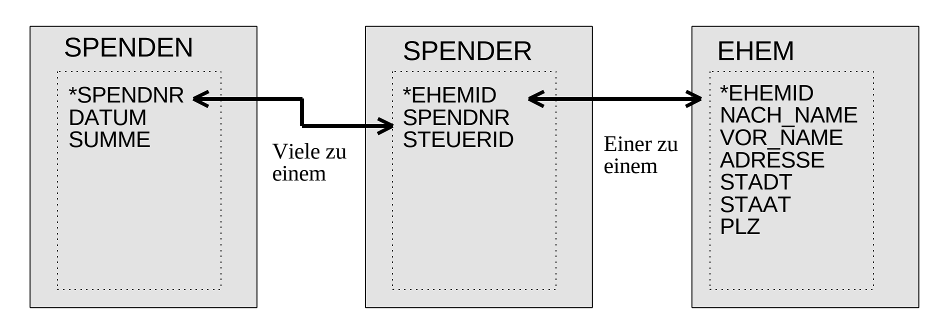


Abb. 1.6: Eine Relation planen

Die Indizes identifizieren

Damit CodeReporter Datendateien miteinander verknüpfen kann, müssen Indizes vorhanden sein. Mit Hilfe von Indizes lassen die Datensätze sich rasch lokalisieren. Werden zwei Datendateien miteinander verknüpft, verwendet die erste einen Index, um die entsprechenden Daten in der zweiten Datei zu finden. Indizes werden im allgemeinen zusammen mit der Datendatei erzeugt.

Notieren Sie in der Übersicht über die Relationen die Indizes der einzelnen Datendateien. Hat die Datendatei einen zusammengesetzten Index (der mehr als ein Feld enthält), schreiben Sie den Namen des Indexes unten auf die Feldliste und kennzeichnen diesen als Index. Wenn die Relation auf ein Indexfeld zeigt, machen Sie aus der vorhandenen Linie einen Pfeil.

Zeigt eine Linie in dem Relationsschema auf ein Feld, für das kein Index erstellt worden ist, muß die Relationslinie möglicherweise entfernt bzw. mit einem anderen Datenwerkzeug ein Index für das Feld erstellt werden.

Damit eine Datendatei in die Relation aufgenommen wird, braucht lediglich eine einzige Verknüpfung vorhanden zu sein.

Die Top-Master-Datendatei bestimmen

Nach Fertigstellung der Übersicht sollten Sie den für die Relation besten Ablauf bestimmen. Ordnen Sie die Datendateien in einem Baumdiagramm an, wobei eine Datei an der Spitze steht; die übrigen Dateien nehmen ihre von den Verbindungslinien vorgegebenen Positionen ein.

Bei vielen Relationen kann man die Datendateien auf verschiedene Art und Weise zusammenstellen. Berücksichtigen Sie folgende Richtlinien, um den besten Ablauf für eine Relation zu bestimmen:

- Für jede untergeordnete Datendatei muß ein Subindex vorhanden sein, auf den die nächsthöhere Datendatei zeigt.
- Viele-zu-einem- und Einer-zu-einem-Relationen eignen sich besser als Einer-zu-vielen- und Viele-zu-vielen-Relationen.

Aus Abb. 1.7 gehen die verschiedenen Baumdiagramme hervor, die nach dem Relationsschema in Abb. 1.6 möglich sind. Die erste Relation enthält keine Einer-zu-vielen-Relationen und kann aufgrund dessen, rein theoretisch, besser ablaufen als die beiden anderen. Natürlich kann die Beschaffenheit des Reports die Verwendung eines der anderen Relationsbäume notwendig machen.
Ist im Report z. B. eine Liste aller Ehemaligen (nicht nur derjenigen, die Beiträge leisten) erforderlich, müßten Sie die Relation verwenden, bei der EHEM oben

steht. Im Kapitel „Relationale Reports“ beschreiben wir, wie sich die Beziehungen einer Relation auf das Ergebnis des Reports auswirken können.

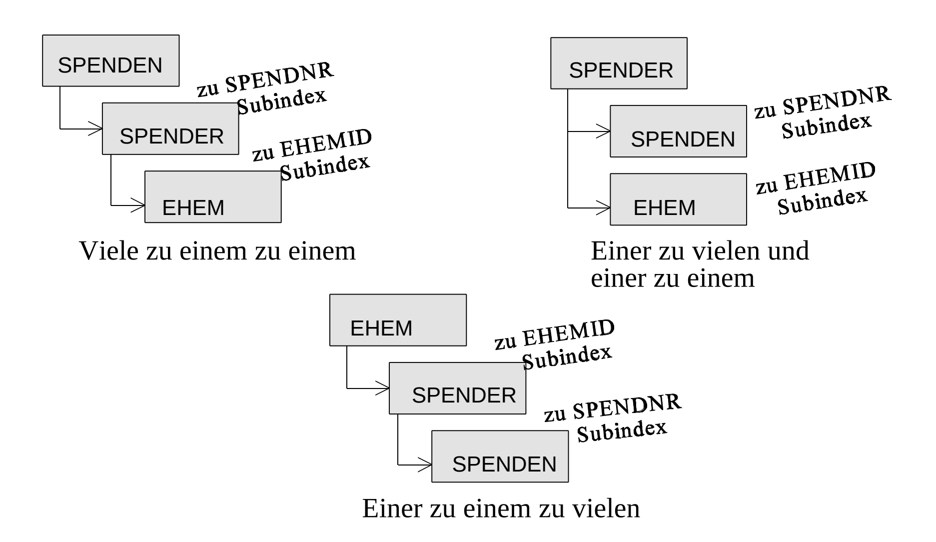


Abb. 1.7: Relationsbäume

Den Report gestalten

Nachdem ein Schema wie in Abb. 1.7 erstellt worden ist – vorausgesetzt, alle für den Report erforderlichen Datendateien können miteinander verbunden werden –, kann man mit der Gestaltung des Reports beginnen. Auf den restlichen Seiten des vorliegenden Handbuchs erläutern wir detailliert die Erstellung eines relationalen Reports.

Den Report auf Stichhaltigkeit überprüfen

Wenn Sie den Report, wie im vorliegenden Kapitel erläutert, mit seinen Bereichen, Relationen und Elementen entworfen haben, sollten Sie ihn noch einmal anhand der Zweckerklärung überprüfen, ob er letztendlich den darin definierten Anforderungen gerecht wird. Besteht er diese letzte Prüfung, können Sie die Entwurfsphase als abgeschlossen betrachten.

2 Relationale Reports

Bei einem relationalen Report handelt es sich um einen Report, der auf mehrere Datendateien zugreift. Diese werden in den Report integriert, um eine zusammenhängende „zusammengesetzte Datendatei“ als Ausgangsbasis für den Report zu bilden. Im allgemeinen ziehen relationale Reports Informationen aus mehreren verknüpften Datendateien und präsentieren sie in einer lesbaren Art und Weise.

Relationen

Bei der Erstellung eines relationalen Reports, der auf mehrere Datendateien zugreift, müssen die entsprechenden Informationen in den Datendateien genau lokalisiert werden. Das heißt, die Daten, die ein Datensatz einer Datei enthält, müssen logisch mit den Datensätzen der anderen Datendateien übereinstimmen. Z. B. sollte die Rechnung für den Kunden mit der Nr. 2345, Jane Smith, nicht an den Kunden mit der Nr. 4321, Peter Rodriguez, geschickt werden.

Um sicherzustellen, daß innerhalb des Reports die richtigen Daten aus allen Datendateien abgerufen werden, muß die Art und Weise der Informationsbeschaffung beim Entwerfen des Reports klar beschrieben werden. Ist diese Beschreibung vorhanden, sind die Datendateien verbunden oder verknüpft. Eine Relation besagt z. B.: „Benutzt der Report einen Datensatz aus *dieser* Datendatei, soll *diese* Information daraus verwendet werden, um einen Datensatz mit *derselben Information* in *jener* Datendatei zu lokalisieren.“

Die Datendateien stehen in einem hierarchischen Master-Slave-Verhältnis zueinander. Die steuernde Datendatei wird als Master, die verknüpfte (in der die Suchvorgänge durchgeführt werden) als Slave bezeichnet.

Zerlegen wir die oben angegebene Relation in ihre Bestandteile, treten die einzelnen Elemente klarer hervor.

- „Benutzt der Report einen Datensatz aus dieser Datendatei“ – legt die Masterdatei fest.
- „soll diese Information daraus verwendet werden“ – ist ein dBASE-Ausdruck, der für jeden Datensatz der Masterdatei ausgewertet wird. Dieser Ausdruck, der sogenannte Masterausdruck, dient als Suchschlüssel, der den Slave-Subindex durchsucht.

- „um einen Datensatz mit derselben Information in jener Datendatei zu lokalisieren“ – bezeichnet die für den Suchvorgang verwendete Slavedatei und deren Indexfolge, aufgrund derer der entsprechende Datensatz lokalisiert wird. Die in der Relation verwendete Indexfolge wird auch als Slave-Subindex bezeichnet.

Eine Relation zwischen zwei Datendateien ist lediglich in einer Richtung wirksam; aus einer Datendatei werden Informationen in einer anderen gesucht. Eine Datendatei steuert die zweite, deren Datensätze nur auf Anweisung der ersten abgerufen werden.

Mit anderen Worten: eine Datendatei enthält Informationen, die Verweise auf Informationen in einer zweiten Datendatei liefern. Sind die beiden Dateien verknüpft, weist der Master (die erste Datendatei) die Slavedatei an, einen Datensatz im Slave zu lokalisieren, der dieselben Daten enthält. Diese Daten kann die Masterdatei anschließend verwenden und ausgeben; dabei wird nicht einfach auf die Daten in der Slavedatei verwiesen, sondern diese werden tatsächlich gebraucht.

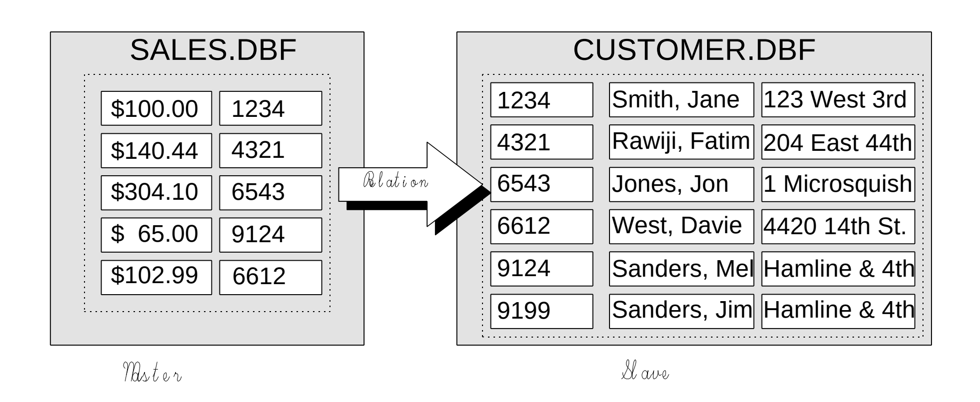


Abb. 2.1: Eine einfache Relation

Wie könnte ein Report, der die Verkäufe auflistet, auch die Käufer aufführen, wenn zwischen SALES.DBF und CUSTOMER.DBF keine Relation bzw. Verknüpfung hergestellt worden ist?

Ist eine Relation vorhanden, kann man die Felder aller Datendateien als Bestandteile einer einzigen Datei, einer sogenannten zusammengesetzten Datendatei betrachten. Diese Datei ist in Wirklichkeit nicht auf der Platte vorhanden – CodeReporter richtet physisch keine Datei ein, die alle Daten aus allen verknüpften Datendateien enthält –, sondern ist das Ergebnis der komplizierten Methode, wie CodeReporter die Daten in den einzelnen Datendateien verwaltet.

Nachdem die Relation hergestellt worden ist, umfaßt die zusammengesetzte Datendatei die Felder aller in der Relation vorhandenen Datendateien. Die in den Feldern der zusammengesetzten Datendatei enthaltenen Daten aktualisiert CodeReporter während der Reportausgabe.

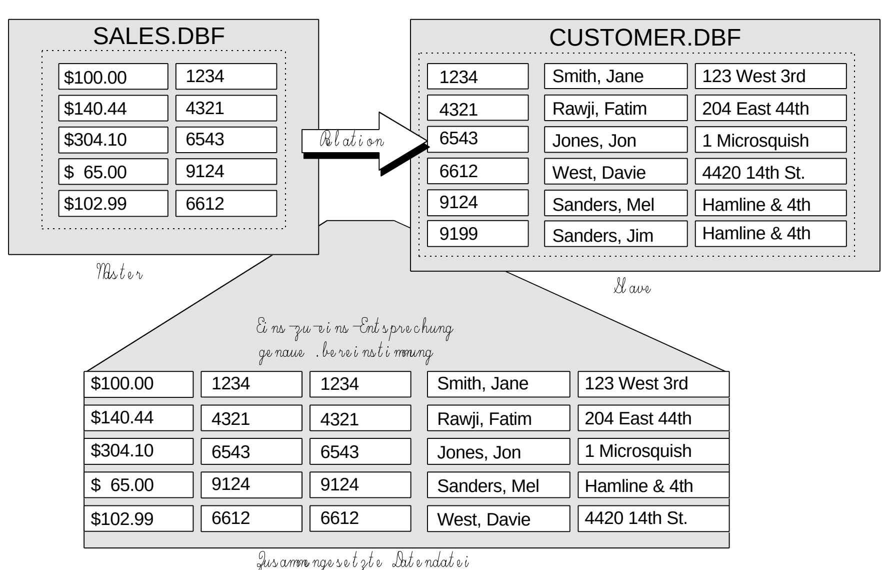


Abb. 2.2: Zusammengesetzte Datendatei
Obgleich die zusammengesetzte Datendatei die Felder der Masterdatei enthält, müssen die der Slavedatei notwendigerweise nicht ebenfalls vorhanden sein.

Hierbei handelt es sich um eine unmittelbare Auswirkung der Master-Slave-Beziehung. Bei Einsatz der Relation werden die Datensätze der Masterdatei zur Lokalisierung entsprechender Datensätze in der Slavedatei verwendet. Allerdings sind für die Masterdatei keine Verweise auf alle Datensätze der Slavedatei erforderlich; letztere wird lediglich für Suchoperationen gebraucht.

Die Schritte, die CodeReporter dabei intern unternimmt, sind:

1. Auf einen Datensatz in der Masterdatei zeigen.
2. Die gemeinsamen Daten in der Masterdatei übernehmen (vom Masterausdruck definiert).
3. Einen Datensatz mit denselben Daten in der Slavedatei suchen (bildet jetzt den zusammengesetzten Datensatz).
4. Schritte 1–3 wiederholen, bis alle Datensätze der Masterdatei durchlaufen worden sind.

Enthält die Masterdatei keinen Verweis auf einen bestimmten Datensatz in der Slavedatei, wird dieser offensichtlich nicht in die zusammengesetzte Datendatei übernommen. Aus diesem Grund ist es in der Reportentwurfsphase besonders wichtig, die Zweckerklärung zu berücksichtigen. Entscheiden Sie sich für eine falsche Top-Master-Datendatei, kann das tatsächliche Ergebnis des Reports erheblich von dem gewünschten abweichen.

Das ist in der zusammengesetzten Datendatei in Abb. 2.2 der Fall. CUSTOMER.DBF enthält einen Eintrag für Jim Sanders (Kunden-Nr. 9199). Doch weil diese Kundennummer in der Masterdatei SALES.DBF nicht vorkommt, wird Jim Sanders nicht in die zusammengesetzte Datendatei übernommen.

Auch kann es vorkommen, daß die Slavedatei keinen der Masterdatei entsprechenden Datensatz aufweist. Letztere fordert Daten aus der Slavedatei an, doch diese kann die Informationen nicht zur Verfügung stellen.

Wenn eine solche Situation auftritt, kann CodeReporter – abhängig davon, wie die Relation gebildet wurde – auf dreierlei Weise reagieren.

1. Felder löschen. CodeReporter füllt die Felder der Slavedatei mit Leerstellen auf; numerische Felder erhalten den Wert Null. Hierbei handelt es sich um die Voreinstellung.
2. Nächster Satz. Der zusammengesetzte Datensatz wird ignoriert. Sowohl Master- als auch Slavedatensatz werden übersprungen, als seien sie nicht vorhanden.

3. <u>Stop bei Fehler</u>. Der Report wird unterbrochen, und es erscheint eine Fehlermeldung. Diese Fehlerbehandlung ist angebracht, wenn Daten in der Slavedatei lokalisiert werden müssen; fehlen diese Daten, wird der Report abgebrochen.

Komplexe Relationen

Im Prinzip reichen zwei Datendateien aus, wenn sie alle für den Report erforderlichen Daten enthalten. Werden für einen Report allerdings drei oder mehr Datendateien gebraucht, muß eine komplexe Relation mit darin enthaltenen Subrelationen her. Das heißt, der Slave einer Datendatei fungiert als Master für einen Slave auf einer niedrigeren Stufe.

Eine Masterdatei kann unbegrenzt viele Slavedateien haben, und eine Slavedatei wiederum kann für andere Slavedateien die Masterdatei sein, um weitere Relationen herzustellen. Dieses Phänomen erlaubt die Bildung eines „Relationsbaumes“, bei dem alle Relationen von einer Masterdatei ausgehen.

Hinweis: An dieser Stelle kann unsere Begriffswahl leicht zu Verwirrungen führen. Der Begriff „Top-Master-Datendatei“ bezieht sich per Definition auf die oberste Datendatei des Relationsbaumes, die gleichzeitig die Primärdatei des Reports ist. Der Begriff „Masterdatei“ dagegen bezieht sich auf die steuernde Datendatei derjenigen Relation, mit der wir uns zur Zeit beschäftigen.

Alle Relationen, die von der Top-Master-Datendatei ausgehen, werden insgesamt als Relationsset bezeichnet. Abb. 2.3 stellt eine komplexe Relation dar; die Datei STUDENT.DBF ist der Master der Datei ENROLL.DBF, die wiederum der Master der Datei COURSE.DBF ist.

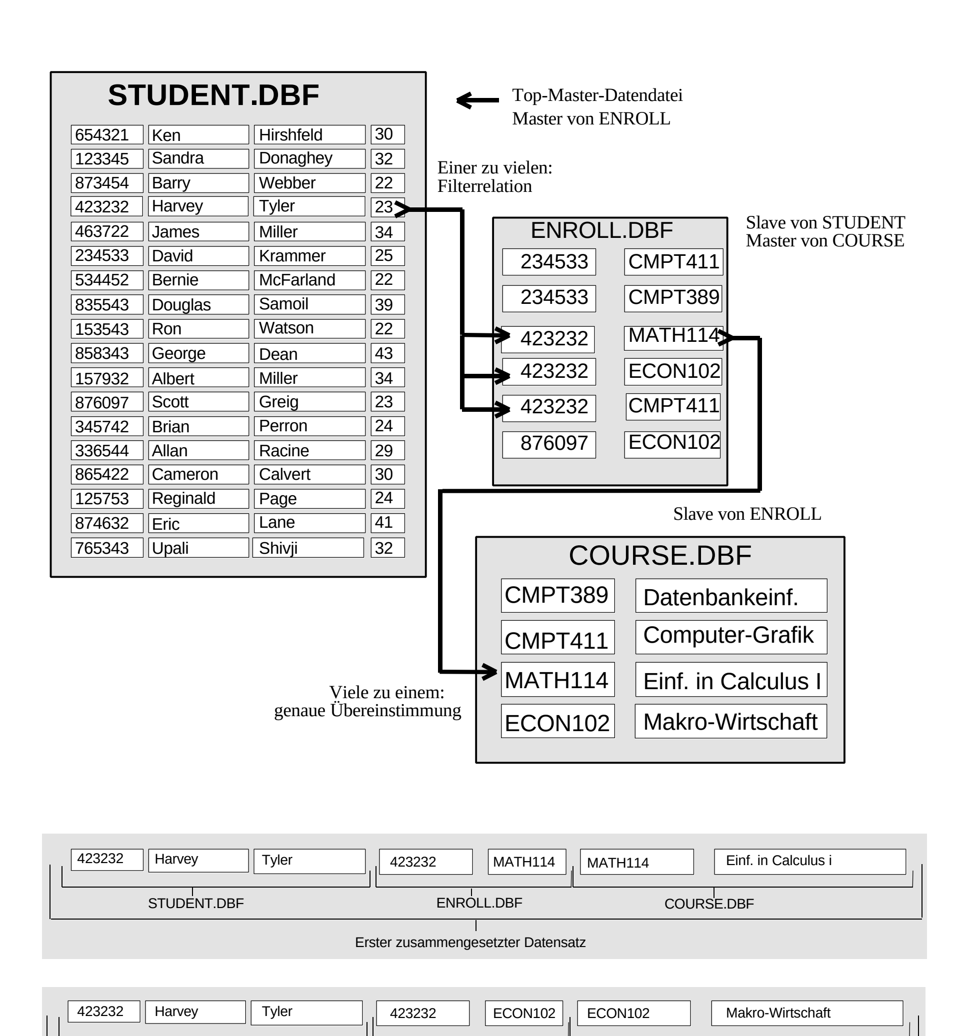


Abb. 2.3: Komplexe Relation

Relationstypen

CodeReporter unterstützt drei verschiedene Arten von Relationen, die jeweils auf unterschiedliche Weise auf die Daten der Slavedatei zugreifen. Die verfügbaren Relationstypen sind Relationen mit genauer Übereinstimmung, Filterrelationen und Relationen mit ungefährer Übereinstimmung.

Relationen mit genauer Übereinstimmung

Eine Relation mit genauer Übereinstimmung läßt zwischen der Master- und der Slavedatei lediglich eine Übereinstimmung zu, das heißt, daß jeder Datensatz der Masterdatei nur auf einen Datensatz in der Slavedatei paßt. Bei einer Relation mit genauer Übereinstimmung wird, auch wenn mehrere Datensätze auf den ausgewerteten Masterausdruck passen, lediglich der erste passende Datensatz zurückgegeben. Das ist eine Einer-zu-einem-Relation; ein Datensatz in der Masterdatei lokalisiert einen Datensatz in der Slavedatei.

Mit anderen Worten: CodeReporter durchsucht für jeden Datensatz der Masterdatei den Index der Slavedatei, bis es auf einen Subindexeintrag stößt, der denselben Inhalt wie der ausgewertete Masterausdruck hat. Die Suche endet mit dem ersten übereinstimmenden Datensatz bzw. nach dem Durchlaufen des gesamten Slave-Subindexes.

Hinweis: Im allgemeinen läßt sich eine Relation mit genauer Übereinstimmung am besten in Verbindung mit einem eindeutigen Subindex verwenden.

Abb. 2.2 zeigt eine Relation mit genauer Übereinstimmung zwischen SALES.DBF und CUSTOMER.DBF.

Relationen mit ungefährer Übereinstimmung

Beim zweiten Relationstyp geht es um ungefähre Übereinstimmung. Dieser Typus ähnelt der Relation mit genauer Übereinstimmung insofern, als für einen Masterdatensatz lediglich eine Übereinstimmung zugelassen wird. Der einzige Unterschied besteht im Verhalten der Relation, wenn die Slavedatei keinen genau passenden Datensatz enthält. Kommt es zu keiner genauen Übereinstimmung, wird statt dessen der Slavedatensatz genommen, dessen Indexschlüssel als nächster im Slave-Subindex steht.

Relationen mit ungefährer Übereinstimmung sind im allgemeinen eher selten und werden nur benutzt, wenn innerhalb von Datensätzen ein Bereich durch einen einzigen hohen Wert dargestellt werden soll.

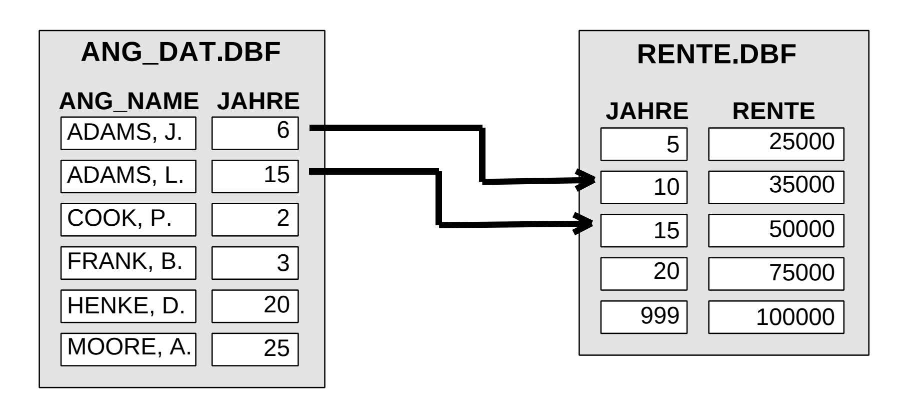


Abb. 2.4: Relation mit ungefährer Übereinstimmung

Abb. 2.4 zeigt eine Relation mit ungefährer Übereinstimmung. In diesem Fall richten sich die Rentenboni für die Angestellten nach der Anzahl der Jahre, die sie für die Firma gearbeitet haben. Statt eines Eintrags für jedes mögliche Dienstjahr enthält die Datei RENTE.DBF lediglich die Obergrenze für die einzelnen Auszahlungsstufen. Die erste Stufe reicht von null bis fünf Jahre, die zweite von sechs bis zehn usw., bis zur höchsten Stufe von 21 Jahren und darüber, für die 100 000 ausgeschüttet werden.

Hinweis: Wird bei einer Relation mit ungefährer Übereinstimmung keine genaue Übereinstimmung gefunden, muß der Datensatz aus der Slavedatei nicht notwendigerweise dem des Masterausdrucks am nächsten kommen. Es kann sich durchaus um den ersten Datensatz handeln, dessen Schlüsselwert größer ist als der Masterausdruck.

In Abb. 2.4 findet der Masterausdruck mit dem Wert sechs den Subindexeintrag für zehn, obgleich sechs als Zahl der Fünf näherkommt.

Filterrelationen

Bei Filterrelationen besteht eine Einer-zu-vielen-Übereinstimmung zwischen der Master- und der Slavedatei. Das heißt, daß CodeReporter für jeden Datensatz der Masterdatei alle passenden Datensätze, nicht nur den ersten, in der Slavedatei ausfindig macht.

In Abb. 2.5 wird eine Filterrelation verwendet, um die Masterdatei STUDENT.DBF mit der Datei ENROLL.DBF zu verknüpfen. Diese Relation enthält alle Kurse, für die sich ein Student eingeschrieben hat. Im vorliegenden Beispiel bildet die Filterrelation, ausgehend von einem Datensatz in der Masterdatei, drei zusammengesetzte Datensätze.

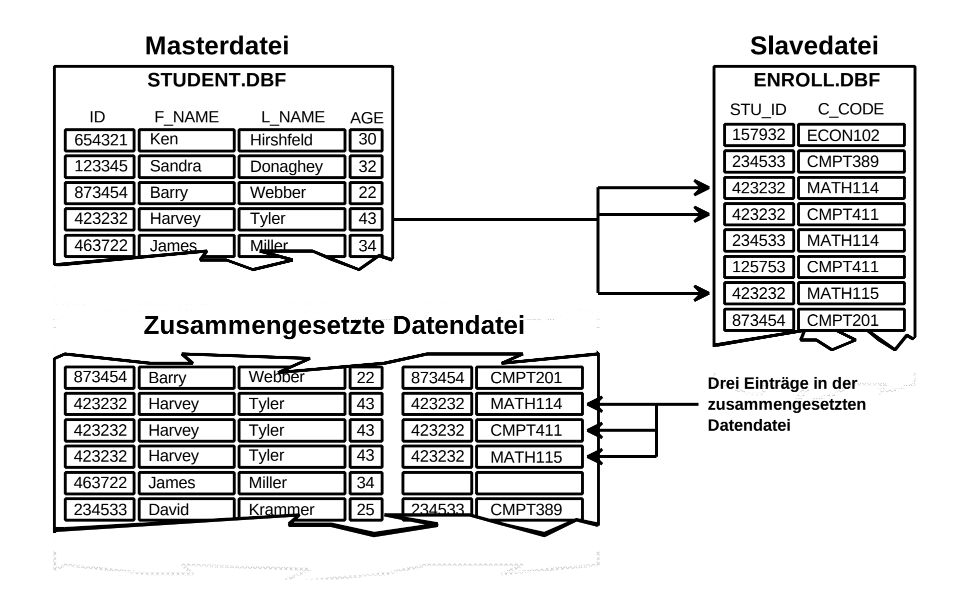


Abb. 2.5: Filterrelation

Masterdatei mit mehreren Slavedateien

Bei einer komplexen Relation ist es auch möglich, zwei oder mehr Slavedateien mit derselben Masterdatei zu verknüpfen. In nahezu jeder Hinsicht stimmt diese Zusammenstellung mit den bereits besprochenen überein – der Masterausdruck

jeder Relation wird für den aktuellen Datensatz der Masterdatei ausgewertet und das Ergebnis als Lookup-Schlüssel für die Slave-Subindizes verwendet.

Ist keine Filterrelation beteiligt, verhält sich die Relation auf die bekannte Art und Weise – der zusammengesetzte Datensatz enthält die Daten aus den Datensätzen der Master- sowie der einzelnen Slavedateien. Dabei spielt es keine Rolle, ob es sich um Relationen mit genauer oder ungefährer Übereinstimmung handelt. Siehe Abb. 2.6.

Ist eine Masterdatei jedoch mit zwei oder mehr Slavedateien per Filterrelation verknüpft, behandelt CodeReporter die Relation etwas anders. Für jede Filterrelation der Masterdatei muß die Fehlerbehandlung auf „Felder löschen“ eingestellt werden; das liegt daran, daß für jede einzelne Filterrelation eine komplexe Relation durchgeführt wird, die für die weiteren Filterrelationen mit Leerzeichen gefüllte Felder ausgibt.

Abb. 2.6 zeigt eine Masterdatei mit zwei Slavedateien, in Relationen mit genauer Übereinstimmung sowie als Filterrelationen.

Abb. 2.6: Komplexe Relation mit einer Master- und zwei Slavedateien

Relationsset

MASTER

AAA	123

SLAVE1

AAA	APFEL
AAA	ORANGE
AAA	PFIRSICH

SLAVE2

123	98765
123	76543
123	32109

Zusammengesetzte Datendateien

als Relationen mit genauer Übereinstimmung

Master		Slave1		Slave2	
AAA	123	AAA	APFEL	123	98765

als Filterrelationen

Master		Slave1		Slave2	
AAA	123	AAA	APFEL		
AAA	123	AAA	ORANGE		
AAA	123	AAA	PFIRSICH		
AAA	123			123	98765
AAA	123			123	76543
AAA	123			123	32109

Achtung! Eine Filterrelation darf bei einer Masterdatei mit zwei oder mehr Slavedateien nicht gleichzeitig mit Relationen mit genauer bzw. ungefährer Übereinstimmung verwendet werden.

Relationen herstellen

Bei der Entwicklung eines Reports sind die darin verwendeten Relationen besonders wichtig. Aus diesem Grund schlagen wir vor, daß Sie sich bei allen Reports, in die mehrere Datendateien eingehen, an die in Kapitel 1 erläuterten Schritte halten und u. a. die Relationen auf dem Papier entwerfen, ehe Sie sie mit CodeReporter herstellen.

Den Top Master angeben

Über die Menüoption **DATEI | NEU** können Sie eine neue Relation erstellen. Die Dialogbox „Datendatei auswählen" wird aufgerufen und erwartet die Eingabe der Top-Master-Datendatei. Lokalisieren Sie den Top Master anhand der Verzeichnis- und Dateiauswahlbox, und klicken Sie auf „OK", um die Datei zu öffnen.

Per Voreinstellung speichert CodeReporter die Pfadnamen der in einem Report verwendeten Datendateien in der Reportdatei mit ab. Beim Laden des Reports werden die entsprechenden Datendateien ebenfalls geladen. Wurden diese in ein anderes Verzeichnis verschoben bzw. gelöscht, kann CodeReporter die Dateien natürlich nicht öffnen. Deshalb sollten Sie in einem solchen Fall die Menüoption **DATEI | ÖFFNEN MIT PFAD** verwenden. Nach der Auswahl einer Reportdatei bittet CodeReporter um die Eingabe des Verzeichnisses, das die Datendateien für den Report enthält. Dieses Verzeichnis wird später zusammen mit dem Report gespeichert.

Hinweis: Haben Sie einen Top Master für den Report bestimmt, kann er nicht mehr gegen einen anderen ausgetauscht werden. Wenn Sie einen neuen Top Master wünschen, müssen Sie mit dem Report noch einmal vollständig von vorn beginnen.

Bitoptimierte Abfragetechnik

CodeReporter bedient sich der bitoptimierten Abfragetechnik (BOA) von Sequiter, um schnelle Abfragen in der zusammengesetzten Datendatei durchzuführen. Dabei wird der Abfrageausdruck mit den Sortierfolgen der Subindizes des Top Masters verglichen. Paßt die Abfrage auf den Subindexausdruck, wird der Subindex selbst dazu verwendet, die Datensätze herauszufiltern, die nicht mit der Abfrage übereinstimmen.

Diese Operation erfolgt selbst bei der größten zusammengesetzten Datendatei blitzschnell, da lediglich die erforderlichen Datensätze physisch von der Platte eingelesen werden.

Wie man die Subindizes der Top-Master-Datendatei öffnet, entnehmen Sie bitte dem Abschnitt „Eine Relation bearbeiten“ weiter unten.

Hinweis: Um einen häufigen Einsatz der BOA zu gewährleisten, empfehlen wir, alle möglichen Subindizes für die Top-Master-Datendatei zu öffnen.

Die Dialogbox „Relation“

Über die Dialogbox „Relation“ (Abb. 2.7), das über die Menüoption **RELATION | BEARBEITEN** aufgerufen wird, lassen sich Slavedateien zum Relationsset hinzufügen.

Diese Dialogbox zeigt die Relation in ihrem Aufbau an; dabei werden bestehende Verknüpfungen durch Linien gekennzeichnet. Die Datendatei links oben ist der Top Master. Werden dem Relationsset weitere Datendateien hinzugefügt, erscheinen sie als Schaltflächen rechts unter der entsprechenden Masterdatei. So bildet sich nach und nach auch optisch der „Relationsbaum“.

Wenn dieselbe Masterdatei mit mehreren Slavedateien verknüpft wird, erscheinen die neuen Schaltflächen unmittelbar unter denen der bereits vorhandenen Slavedateien (siehe Abb. 2.7).

Eine Slavedatei hinzufügen

Eine Slavedatei läßt sich über die Auswahl der Masterdatei und die Menüoption **NEUER SLAVE** bzw. durch Doppelklicken auf die Schaltfläche der Masterdatei zum Relationsset hinzufügen. Dabei spielt es keine Rolle, wenn die Masterdatei bereits die Slavedatei einer anderen Datendatei ist.

Mit der Dialogbox „Datendatei auswählen" wird die Slavedatei aufgesucht und geöffnet. Wählen Sie die Slavedatei wiederum anhand der Verzeichnis- und Dateiauswahlbox aus.

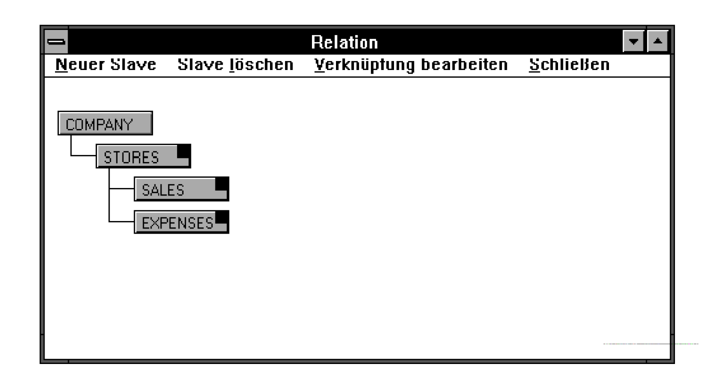


Abb. 2.7: Dialogbox „Relation bearbeiten"

Eine Relation bearbeiten

Die Dialogbox „Datendatei-Verknüpfungen" (Abb. 2.8) wird dazu verwendet, die Relation zwischen einer Master- und einer Slavedatei zu definieren und zu bearbeiten. Diese Dialogbox wird bei der Definition einer neuen Slavedatei automatisch aufgerufen. Später läßt sich die Box zwecks Bearbeitung der Relation nach der Markierung des entsprechenden Slavedateikästchen über die Menüoption **VERKNÜPFUNG BEARBEITEN** bzw. durch Klicken der rechten Maustaste auf die Schaltfläche der Slavedatei wieder aufrufen.

Handelt es sich bei der ausgewählten Dateischaltfläche um den Top Master der Relation und aktivieren Sie **VERKNÜPFUNG BEARBEITEN** (oder klicken die rechte Maustaste), wird die Dialogbox „Master-Indexdateien" aufgerufen. Mit Hilfe dieser Dialogbox können Sie Indexdateien für den Top Master öffnen und/oder schließen – und so die BOA für den Report ermöglichen.

Ist die Relation neu, enthält die Dialogbox „Datendatei-Verknüpfungen" die voreingestellten Informationen, wie aus Abb. 2.8 hervorgeht. Ehe die neue Relation gültig werden kann, müssen Sie das Editierfeld „Masterausdruck" bearbeiten und einen vorhandenen Subindex auswählen.

Der Masterausdruck ist ein auf der Masterdatei (bzw. auf Datendateien auf höherer Stufe) basierender dBASE-Ausdruck, der als Lookup-Schlüssel den Slave-Subindex durchsucht. Bei Durchführung des Reports wird dieser Ausdruck für jeden aus der Masterdatei übernommenen Datensatz ausgewertet; das Ergebnis wird dazu verwendet, über den für die Slavedatei angegebenen Subindex einen Datensatz in der Slavedatei aufzusuchen.

Am häufigsten werden Masterausdrücke verwendet, die nur einen Feldnamen aus der Masterdatei enthalten. Sie können aber auch weitaus komplexer sein und Felder aus höheren Datendateien des Relationsbaumes und/oder dBASE-Funktionen enthalten.

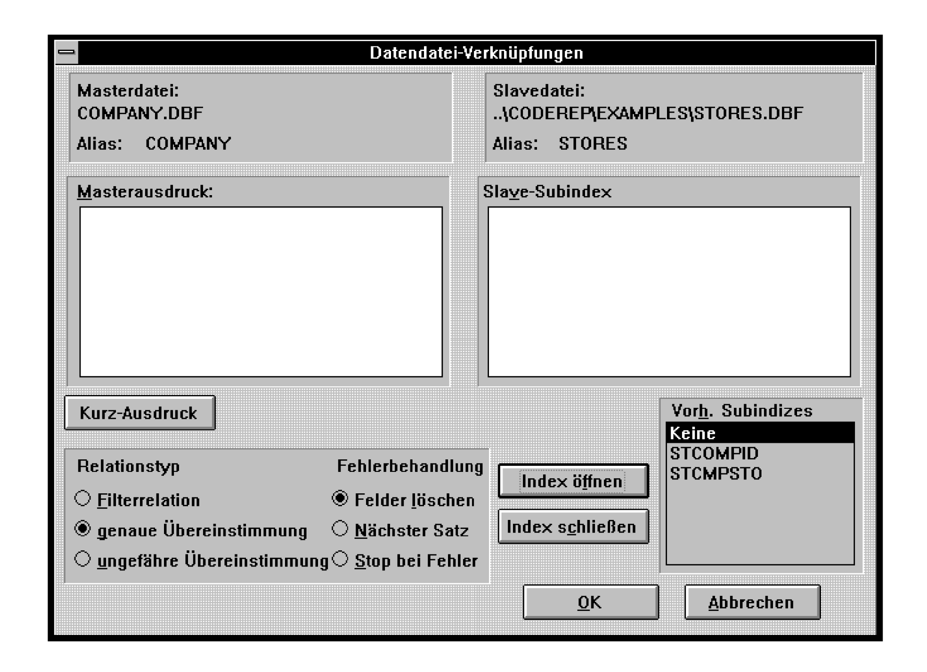


Abb. 2.8: Dialogbox „Datendatei-Verknüpfungen“

Den Masterausdruck können Sie entweder per Hand in das Editierfeld „Masterausdruck“ eingeben oder sich über die Schaltfläche „Kurz-Ausdruck“ die Eingabe des Ausdrucks erleichtern.

Nähere Einzelheiten über dBASE-Ausdrücke und die Dialogbox zur Eingabe von Ausdrücken entnehmen Sie bitte dem Kapitel „Ausdrücke“.

Die Subindizes für die Arbeitsindexdatei der Slavedatei (falls vorhanden) werden in der Listbox „Vorh. Subindizes“ aufgeführt. Wählen Sie einen Subindex aus, so werden Hinweise über ihn, einschließlich der Sortierfolge, im Editierfeld „Slave-Subindex“ angezeigt. Der dBASE-Ausdruck, der für den Subindex aufgeführt wird, sollte, falls in der Relation verwendet, mit dem Masterausdruck für die Lookup-Operation übereinstimmen, damit er richtig funktioniert.

Gibt es keinen dem Masterausdruck entsprechenden Subindex oder verfügt die Slavedatei nicht über einen Arbeitsindex, enthält die Listenbox „Vorh. Subindizes“ lediglich den Eintrag „Keiner“.

Subindizes aus Indexdateien, bei denen es sich nicht um Arbeitsindexdateien handelt, können über die Schaltfläche „Index öffnen“ geöffnet und in der Relation eingesetzt werden. Die Schaltfläche ruft die Dialogbox „Index auswählen“ auf, mit deren Hilfe die neue Indexdatei lokalisiert und ausgewählt werden kann.

Hinweis: CodeReporter arbeitet indexspezifisch und kann lediglich Indexdateien öffnen, die mit dem aktuellen Indexformat kompatibel sind. Innerhalb eines Reports lassen sich die Indexformate nicht nebeneinander benutzen. Das heißt, bei einem anderen Indexformat ist eine indexspezifische Version von CodeReporter erforderlich. Im Kapitel „Starten mit CodeReporter“ finden Sie Hinweise auf die Verwendung der entsprechenden CodeReporter-DLLs.

Haben Sie eine Indexdatei geöffnet, können deren Subindizes für die Relation in der Listbox „Vorh. Subindizes“ ausgewählt werden.

Indexdateien mit nicht verwendeten Subindizes lassen sich über die Schaltfläche „Index schließen“ schließen. Nach der Auswahl wird die mit dem ausgewählten Slave-Subindex verbundene Indexdatei geschlossen, und alle anderen Subindizes für den Index werden entfernt. Da zusätzlich geöffnete Indexdateien sich aber nicht negativ auf die Programmausführung auswirken, braucht man sie nicht unbedingt zu schließen.

Für Lookup-Operationen erlaubt CodeReporter ebenfalls eine andere Methode, bei der keine Subindizes aus der Slavedatei verwendet werden. Der Masterausdruck kann neben dem Lookup-Schlüssel auch eine Datensatznummer als Ergebnis haben. Diese Datensatznummer bezeichnet die physische Nummer des aus der Slavedatei stammenden Datensatzes.

Dieser Methode bedient man sich in erster Linie bei statischen, unveränderlichen Slavedateien, die nie gepackt werden. Der Vorteil besteht darin, daß Suchoperationen rascher und effizienter durchgeführt werden als bei einem Subindex.

Wählen Sie zur Durchführung einfach die Option „Keiner“ in der Listbox „Vorh. Subindizes“ an. Es wird kein Subindex ausgewählt, und der Masterausdruck wird als Datensatznummer übernommen.

Der Entwickler des Reports ist dafür verantwortlich, daß die Datensatznummer für den ausgewerteten Masterausdruck auf die Datensatznummer der Slavedatei verweist, die dem Datensatz der Masterdatei entspricht.

Hinweis: Enthält der ausgewertete Masterausdruck einen Verweis auf eine nicht vorhandene Datensatznummer (<= 0 oder > Anzahl der Slavedatensätze), generiert CodeReporter bei der Erstellung des Reports einen Fehler mit der Nummer -70.

Die Dialogbox „Datendatei-Verknüpfungen“ wird auch dazu benutzt, den Typ der Relation von Master- zu Slavedateien einzustellen. Die einzelnen Relationstypen bestimmen, wie Datensätze aus der Slavedatei abgerufen werden. Nähere Informationen zu den verschiedenen Relationstypen finden Sie weiter oben.

Der voreingestellte Relationstyp „genaue Übereinstimmung“ kann mit Hilfe der Optionsschaltflächen „Filterrelation“ oder „ungefähre Übereinstimmung“ auf einen neuen Wert gesetzt werden.

Die Optionsschaltflächen unter dem Begriff „Fehlerbehandlung“ steuern das Verhalten von CodeReporter, wenn bei einer Lookup-Operation in einer Slavedatei keine Datensätze gefunden werden. Nähere Informationen zur Fehlerbehandlung finden Sie weiter oben.

Anstelle der voreingestellten Fehlerbehandlung „Felder löschen“ können Sie die Optionsschaltfläche „Nächster Satz“ bzw. „Stop bei Fehler“ aktivieren.

Hinweis: Bei Relationen mit ungefährer Übereinstimmung ist als einzige Fehlerbehandlung „Felder löschen“ möglich.

Wie aus Abb. 2.7 ersichtlich, beinhalten die Schaltflächen der Datendateien in der Dialogbox „Relation“ Henkel zum Verschieben. Mit Hilfe eines Mausklicks auf

das dunkelgraue Quadrat in der oberen rechten Ecke der Schaltfläche kann eine Slavedatei innerhalb derselben Masterdatei versetzt werden.

Der einzige Vorteil, eine Slavedatei in bezug auf ihre Masterdatei höher oder tiefer zu setzen, besteht darin, daß niedrigere Slavedateien die Felder von höheren Slavedateien im Masterausdruck für die Relation benutzen können.

In Abb. 2.7 kann die Relation STORES–EXPENSES zur Definition der Relation alle Felder der Dateien COMPANY.DBF, STORES.DBF und SALES.DBF benutzen. Die Relation STORES–SALES andererseits kann in ihrem Masterausdruck lediglich die Felder von COMPANY.DBF und STORES.DBF verwenden.

In bezug auf ihre Masterdatei kann eine Datendatei durch Klicken der linken Maustaste auf den Henkel und Ziehen mit der Maus versetzt werden. Darf die Datei nicht an eine andere Stelle gesetzt werden, hat das Anklicken und Ziehen keine Auswirkung.

Die zusammengesetzte Datendatei sortieren

Die Eingabe von Daten in eine Datendatei erfolgt gewöhnlich ungeordnet. Die Verkäufe an den Kunden Nr. 1 werden in ihrer Gesamtheit nicht immer vor den Verkäufen an den Kunden Nr. 300 eingegeben. Bei Reports ist es jedoch im allgemeinen üblich, die Daten in einer logischen Reihenfolge auszugeben – alphabetisch, numerisch, nach Datum usw.

Mit CodeReporter können Sie die zusammengesetzte Datendatei über einen Sortierausdruck sortieren lassen. Dieser dBASE-Ausdruck wird für jeden zusammengesetzten Datensatz ausgewertet; das Ergebnis (und die zusammengesetzten Datensätze) wird in der neuen logischen Reihenfolge eingelesen. Diesen Vorgang bezeichnen wir als das Sortieren der zusammengesetzten Datendatei.

Der Sortierausdruck

Der Sortierausdruck besteht einfach aus einem dBASE-Ausdruck, der ein Feld oder eine Kombination von Feldern aus der zusammengesetzten Datendatei enthalten kann. In Abb. 2.9 sind die Auswirkungen eines Sortierausdrucks zu sehen. In den Sortierausdruck eingeschlossen ist jeweils ein Feld aus Master- und Slavedatei.

Der Sortierausdruck wird in die Dialogbox „Ausdruck eingeben“ eingegeben. Diese Box können Sie über die Menüoption **REPORT | SORTIERAUSDRUCK** auf rufen.

Einzelheiten zu dBASE-Ausdrücken und der Dialogbox „Ausdruck eingeben“ finden Sie im Kapitel „Ausgabeobjekte“ unter der Überschrift „Ausdrücke“.

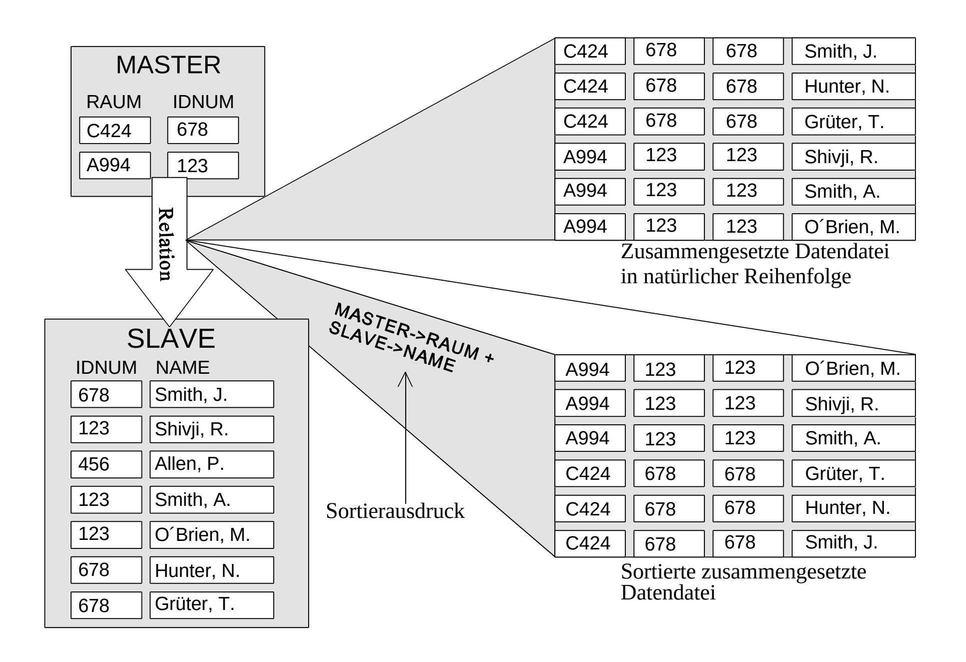


Abb. 2.9: Ein Relationsset sortieren

Die zusammengesetzte Datendatei abfragen

Eine Abfrage ist ein logischer Ausdruck, mit dessen Hilfe eine Untermenge der zusammengesetzten Datendatei angegeben wird. Dieser Abfrageausdruck bildet einen Filter, durch den die zusammengesetzten Datensätze der Relation geschickt werden. In den Report werden lediglich die Datensätze übernommen, die das Filterkriterium erfüllen. Das heißt, der Ausdruck wird für jeden zusammengesetzten Datensatz der zusammengesetzten Datendatei ausgewertet; ist das Ergebnis logisch wahr, erscheint der Datensatz im Report, ansonsten wird er ignoriert.

Sollen zum Beispiel in die Relation in Abb. 2.9 lediglich die Personen in Raum A994 eingeschlossen werden, deren Namen mit einem „S“ beginnen, sähe die abgefragte zusammengesetzte Datendatei (in natürlicher Reihenfolge) aus, wie in Abb. 2.10 angegeben.

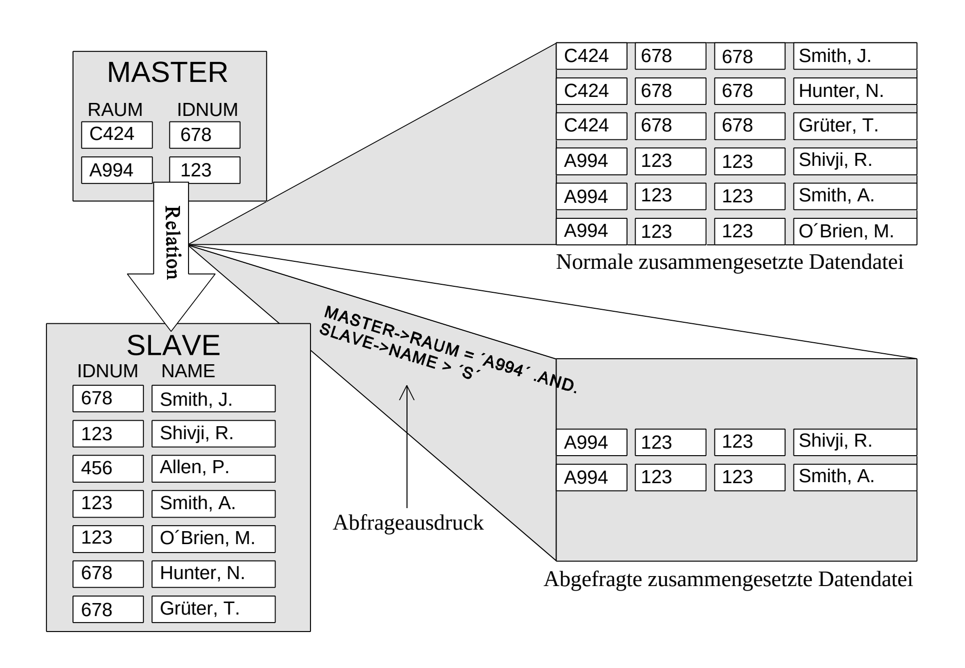


Abb. 2.10: Abgefragte Datendatei

Der Abfrageausdruck wird in die Dialogbox „Ausdruck eingeben“ eingegeben. Die Box können Sie über die Menüoption **REPORT | ABFRAGEAUSDRUCK** aufrufen.

Relationen auf Platte

Das Relationsset eines Reports entspricht bzw. ähnelt häufig dem anderer Reports. Die von CodeReporter angelegten Relationssets können Sie auf der Platte ablegen und mit den Menüoptionen **DATEI | RELATION SPEICHERN** und **DATEI | RELATION LADEN** in einen neuen Report einlesen. Diese Menüoptionen bitten Sie um die Eingabe eines Namens für die Reportdatei und speichern/laden die Relation.

Die Relationen werden in besonderen CodeReporter-Relationsdateien mit der Erweiterung .REL abgelegt. Soll eine Relation nicht in weitere Reports eingehen, braucht sie nicht gespeichert zu werden, da die CodeReporter-Reportdatei (.REP) die Relation des Reports enthält.

Achtung! Wenn Sie eine Relation von der Platte laden, wird die aktuelle Relation des Reports sowie alle Objekte, Summen und Berechnungen gelöscht.

Als Code speichern

CodeReporter kann eine Relation auch als Quellcode speichern, der mit den API-Funktionsaufrufen verarbeitet werden kann. Verwenden Sie für diesen Fall die Menüoption **RELATION | ALS CODE SPEICHERN**. Nachdem Sie den Dateityp ausgewählt haben, geben Sie in der Dialogbox „C-Quelldatei angeben“ (Abb. 2.11 unten) einen Dateinamen und ein Verzeichnis ein. Diese Datei kann dann mit dem CodeReporter-API verwendet werden.

Im Anhang F „CodeBasic-API“ finden Sie weitere Hinweise auf die Verwendung der generierten Quellcodedatei.

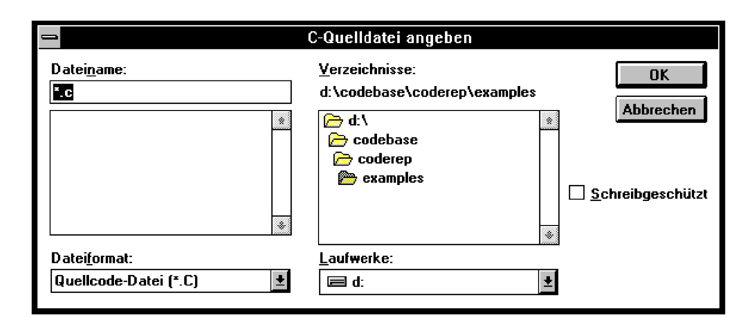


Abb. 2.11: Dialogbox „Als Code speichern“

Beispiel

Um den Aufbau einer Relation an einem Beispiel zu verdeutlichen, erläutern wir in diesem Abschnitt die für den Aufbau der in Abb. 2.7 angegebenen Relation erforderlichen Schritte.

Top-Master-Datendatei

Beim Anlegen einer neuen Reportdatei wird die Top-Master-Datendatei, in unserem Fall COMPANY.DBF, für alle Relationen bestimmt. Eine neue Reportdatei legen Sie mit **DATEI | NEU** an. Markieren Sie nach dem Erscheinen der Dialogbox „Datendatei auswählen“ die Datei COMPANY.DBF (im CodeReporter-Beispieleverzeichnis .\EXAMPLES), und klicken Sie auf „OK“.

Der Report-Entwicklungsbildschirm wird angezeigt. Wählen Sie **RELATION | BEARBEITEN** aus, um die Slavedateien zum Relationsset hinzuzufügen. Die Dialogbox „Relation“ sieht aus, wie in Abb. 2.12 angegeben.

Die Top-Master-Datendatei wird als Schaltfläche mit dem Namen der Datendatei angezeigt. Die nächste Datendatei, die mit ihr verknüpft werden soll, heißt STORES.DBF. Klicken Sie die Menüoption **NEUER SLAVE** an, und wählen Sie die Datei STORES.DBF in der angezeigten Dialogbox „Datendatei auswählen“ aus.

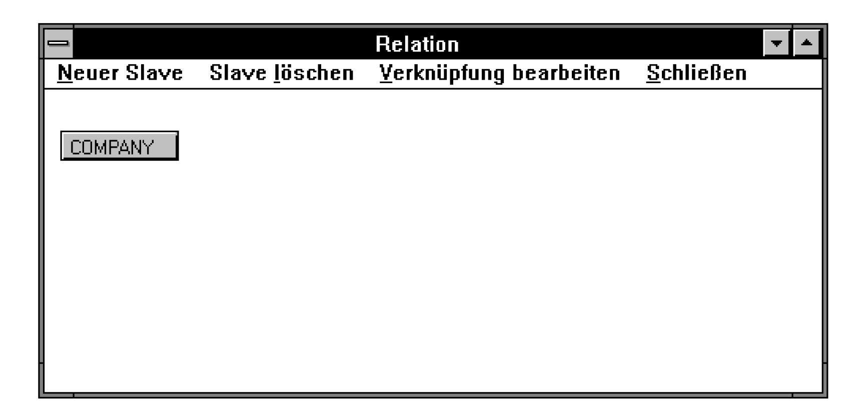


Abb. 2.12: Dialogbox „Relation“

Relation bearbeiten

Da es sich um eine neue Relation handelt, erscheint automatisch die Dialogbox „Datendatei-Verknüpfungen“. In der Listbox „Vorh. Subindizes“ werden für STORES.DBF die Subindizes COMPID (festgelegt auf **`STORES->COMPID`**) und COMP_STORE (festgelegt auf **`STORES->COMPID+STORES->STOREID`**) angezeigt. Da die Masterdatei COMPANY zwar das Feld COMPID, aber nicht das Feld STOREID enthält, sollten Sie den Subindex COMPID von STORES.DBF auswählen.

Betätigen Sie die Tab-Taste auf der Tastatur, bis die Markierung die Listbox „Vorh. Subindizes“ erreicht hat, und wählen Sie daraus COMPID aus, oder klikken Sie den Subindex mit der linken Maustaste an. Im Editierfeld „Slave-Subindex“ werden automatisch Informationen über COMPID angezeigt.

In das Editierfeld „Masterausdruck“ geben Sie einen dem ausgewählten Subindex entsprechenden Masterausdruck ein. Im vorliegenden Fall besteht der Masterausdruck einfach aus einem Feldnamen der Masterdatei. Machen Sie folgende Eingabe (aber betätigen Sie die Eingabetaste noch nicht):

```
COMPANY->COMPID
```

Hinweis: Benutzer von dBASE III und Clipper müssen die Indexdateien mit der entsprechenden Indexsortierung per Hand öffnen. Das erfolgt über die

Schaltfläche „Index öffnen“ in der Dialogbox „Datendatei-Verknüpfungen“. Die bei diesem Beispiel verwendeten Dateien befinden sich im CodeReporter-Beispieleverzeichnis (.\EXAMPLES) und heißen:

- STCOMPID.NTX (.NDX) (für STORES.DBF)
- SACMPSTO.NTX (.NDX) (für SALES.DBF)
- EXCMPSTO.NTX (.NDX) (für EXPENSES.DBF)

Der zwischen COMPANY und STORES bestehende Relationstyp hängt u. a. auch vom Report ab. Da jede Firma in der Regel mehr als eine Niederlassung hat, wird wahrscheinlich eine Filterrelation verwendet. Klicken Sie also auf die Schaltfläche „Filterrelation“.

Wenn Sie nun noch die Standard-Fehlerbehandlung „Felder löschen“ auf „Nächster Satz“ einstellen und auf „OK“ klicken, sieht Ihr Bildschirm aus, wie in Abb. 2.13 angegeben.

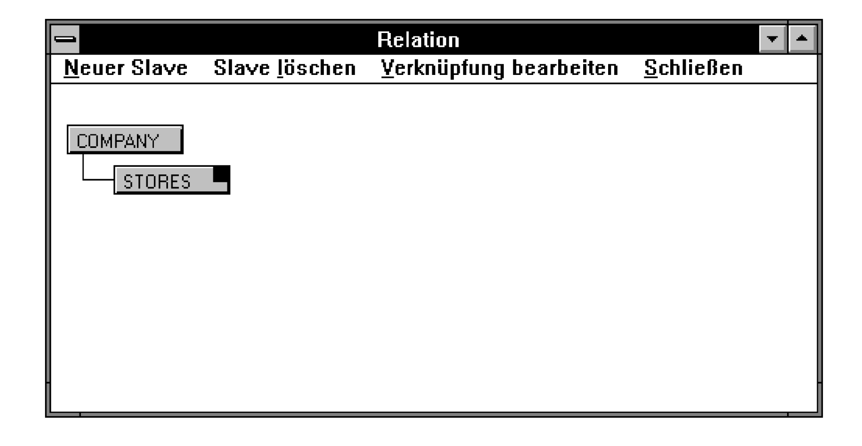


Abb. 2.13: Beispiel für eine noch nicht vollständige Relation

Eine Slavedatei zu einer Slavedatei hinzufügen

Die Datei STORES.DBF ist ein Slave der Datendatei COMPANY.DBF. Damit die Relation ordnungsgemäß Daten für den Umsatz jeder Niederlassung abrufen kann, ist als weitere Datendatei SALES.DBF erforderlich. Diese Datei enthält die Felder COMPID, STOREID, AMOUNT und PRODCODE (einen Verweis auf eine Datendatei mit Produkten). In der Datendatei wird jeder Verkauf firmen- und niederlassungsabhängig geführt. Damit die Verkäufe also den richtigen

Niederlassungen zugeordnet werden können, werden Daten über die Firma und ihre Niederlassung benötigt.

Das heißt, daß die Datendatei SALES.DBF innerhalb der Relation eine Stellung einnehmen muß, von der aus sie sowohl auf die Datei COMPANY.DBF als auch auf STORES.DBF zugreifen kann. Am besten läßt sich SALES.DBF als Slave der Datei STORES.DBF einrichten. Letztere fungiert dann als Slavedatei von COMPANY.DBF und als Masterdatei von SALES.DBF.

Machen Sie mit Hilfe der Tab-Taste STORES.DBF zur aktuellen (zur Zeit COMPANY.DBF) Datendatei, oder klicken Sie STORES.DBF unmittelbar mit der linken Maustaste an.

Um die Datei SALES.DBF zur Relation hinzuzufügen, aktivieren Sie die Menüoption **NEUER SLAVE** bzw. doppelklicken auf die Schaltfläche der Datei. In beiden Fällen wird die Dialogbox „Datendatei auswählen" aufgerufen, in der die Datendatei SALES.DBF markiert und ausgewählt werden kann.

Nach der Dateiauswahl wird die Dialogbox „Datendatei-Verknüpfungen" aufgerufen, um die Relation entsprechend einzurichten. Nehmen Sie folgende Einstellungen vor, und klicken Sie auf „OK":

- Wählen Sie den Subindex COMP_STORE aus.
- Geben Sie als Masterausdruck **`COMPANY->COMPID+STORES->STOREID`** an.
- Aktivieren Sie die Optionsschaltfläche „Filterrelation".
- Vergewissern Sie sich, daß als Standard-Fehlerbehandlung „Felder löschen" eingestellt ist.

Nachdem Sie diese Einstellungen vorgenommen und auf „OK" geklickt haben, wird die Datei SALES.DBF unterhalb von STORES.DBF eingefügt.

Die Positionierung der Datei EXPENSES.DBF erfolgt aufgrund derselben Überlegungen wie bei SALES.DBF. EXPENSES.DBF benötigt Daten aus den Dateien COMPANY.DBF und STORES.DBF, aber nicht aus SALES.DBF. Infolgedessen sollte diese Datei dieselbe Ebene einnehmen wie SALES.DBF.

Aktivieren Sie, während sich die Markierung auf der Schaltfläche von STORES.DBF befindet, die Menüoption **NEUER SLAVE**, wählen Sie die Datendatei EXPENSES.DBF aus, und nehmen Sie folgende Einstellungen in der Dialogbox „Datendatei-Verknüpfungen" vor:

- Wählen Sie den Subindex COMP_STORE aus.
- Geben Sie als Masterausdruck **`COMPANY->COMPID+STORES->STOREID`** an.
- Aktivieren Sie die Optionsschaltfläche „Filterrelation".

- Vergewissern Sie sich, daß als Standard-Fehlerbehandlung „Felder löschen“ eingestellt ist.

Klicken Sie auf „OK“, um die Dialogbox „Datendatei-Verknüpfungen“ zu schließen und zum Dialog „Relation“ zurückzukehren.

Diese komplexe Relation läßt sich nun als Grundlage für die Erstellung eines komplexen Reports nutzen, in den Daten aus allen vier Datendateien einfließen können.

Hinweis: Da diese Relation zwei Filterrelationen beinhaltet, die dieselbe Masterdatei benutzen, können die ausgegebenen Daten geringfügig von der erwarteten Datenausgabe abweichen. Einzelheiten dazu finden Sie unter der Überschrift „Masterdatei mit mehreren Slavedateien“ weiter oben.

3 Gruppen

Was ist eine Gruppe?

Der Sinn und Zweck aller Reports besteht darin, ungeordnet eingegebene Daten in ein übersichtlich strukturiertes Format zu bringen. Ein Report zerlegt diese Daten in logische Abschnitte und/oder faßt sie in einem sinnvollen Format zusammen.

Bei der Sortierung der zusammengesetzten Datendatei werden Datensätze mit gemeinsamen Inhalten nebeneinander gestellt. Die Sortierfolge kann sich auf ein oder mehrere Felder gründen. Das heißt, die zusammengesetzten Datensätze nehmen eine neue Reihenfolge ein, die von den im Sortierausdruck lokalisierten Feldern abhängt. Aufgrund der Sortierung kann man sich die Datensätze auch als logische Untermengen vorstellen.

Eine Datendatei zum Beispiel, die nach Abteilungskürzeln sortiert wird, erzeugt eine zusammengesetzte Datendatei, die Untermengen der verschiedenen Abteilungen enthält. Alle Datensätze, in denen Abteilung „A“ vorkommt, werden vor den Datensätzen mit Abteilung „B“ ausgegeben. Man muß sich die Datensätze mit Abteilung „A“ als Untermenge der zusammengesetzten Datendatei vorstellen.

Enthält der Sortierausdruck mehr als ein Feld der zusammengesetzten Datendatei, enthält die resultierende zusammengesetzte Datendatei sozusagen Oberuntermengen, die ihre eigenen Unteruntermengen enthalten.

Ein Sortierausdruck zum Beispiel, der sowohl das Abteilungskürzel als auch die Lagebeschreibung enthält, bildet einzelne Abteilungsuntermengen, die wiederum ihre eigenen Lageuntermengen enthalten. Abb. 3.1 macht Ober- und Unteruntermengen deutlich.

Also beschreiben diese natürlichen Untermengen ebenfalls den Gesamtablauf des Reports. Der Report geht logisch einen Datensatz nach dem anderen durch und durchläuft jede Ober- und nachfolgende Unteruntermenge der zusammengesetzten Datendatei, bis alle Datensätze verarbeitet worden sind.

Bei der Bearbeitung einer neuen Untermenge der zusammengesetzten Datendatei für den Report besteht jeweils die Möglichkeit, bestimmte Funktionen auszuführen; dazu gehören die Bildung/Zurücksetzung von Summen, die Ausgabe von

Feldern der zusammengesetzten Datendatei, die Ausgabe von Titeln, das Durchführen von Berechnungen usw. Die Funktionen, die bei einem bestimmten zusammengesetzten Datensatz vollzogen werden, richten sich nach der Untermenge, zu der der Datensatz gehört.

Diese untermengenspezifischen Funktionen führt CodeReporter in Konstrukten aus, die man als Gruppen bezeichnet. Am besten lassen sich die Begriffe der Untermenge und Gruppe an einem Beispiel verdeutlichen. Abb. 3.1 gibt eine zusammengesetzte Datendatei wieder, die nach ABT, ORT, ANGEST und DATUM sortiert wird. Wird die Datendatei nach diesen Kriterien sortiert, ergeben sich verschiedene Untermengen – eine für jeden Teil des Sortierausdrucks.

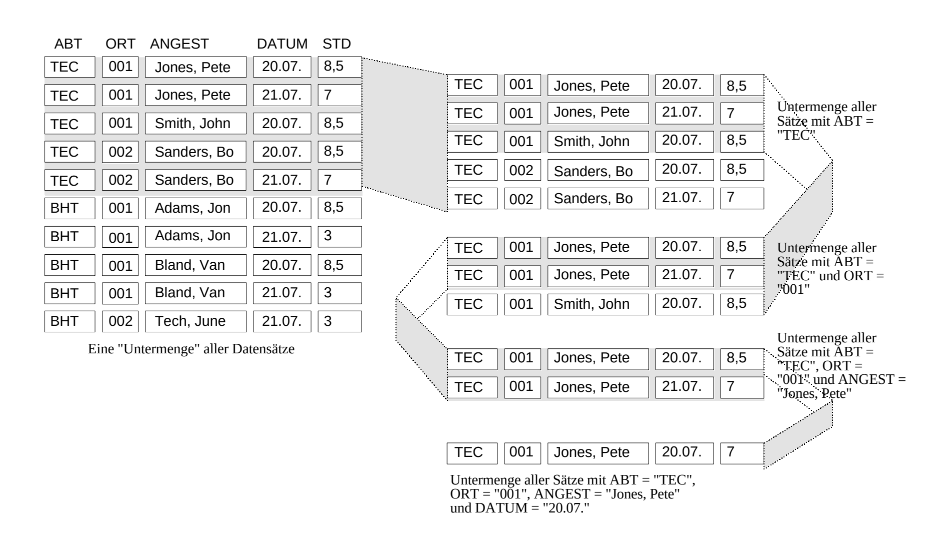


Abb. 3.1: Eine zusammengesetzte Datendatei mit ihren Untermengen

Jeder dieser Untermengen kann eine bestimmte Funktion zugeordnet sein. Dabei kann es sich um eine Teilsumme der Stunden für jede Untermenge, die Gesamtstunden, die jeder Angestellte gearbeitet hat, die Gesamtstunden, die an den verschiedenen Einsatzorten erbracht wurden, handeln usw.

Infolgedessen läßt sich jede Untermenge durch eine Gruppe darstellen, die diese Funktion ausführt. Auch die zusammengesetzte Datendatei läßt sich in diesem Zusammenhang als Gruppe betrachten, da der Report wahrscheinlich die Gesamtsumme aller Abteilungen enthält.

Die sortierte zusammengesetzte Datendatei aus Abb. 3.1 legt die Entwicklung eines Reports mit fünf Gruppen nahe, wie er in Abb. 3.2 dargestellt wird.

Gruppenausdruck

Beim Gruppenausdruck handelt es sich um einen dBASE-Ausdruck, der die Untermenge beschreibt, mit der die Gruppe verbunden ist. Dieser Ausdruck wird für jeden zusammengesetzten Datensatz in der Datendatei ausgewertet; ändert sich der ermittelte Wert, werden die Funktionen der Gruppe ausgeführt (meistens Ausgabe von Gruppenkopf und -fuß für den zusammengesetzten Datensatz). Diesen Wechsel bezeichnen wir als Rücksetzbedingung – oder auch als Gruppenwechsel.

Hat eine Gruppe keinen Gruppenausdruck (d. h., das Editierfeld „Gruppenausdruck“ bleibt leer), besteht für jeden Datensatz der zusammengesetzten Datendatei eine Rücksetzbedingung. Auf diese Weise werden im allgemeinen die genauen Einzelzeilen eines Reports erzeugt.

Achtung! Gewöhnlich enthält ein Report lediglich eine Gruppe ohne Gruppenausdruck. Sind mehrere Gruppen ohne Gruppenausdruck vorhanden, kann das bei der Ausgabe des Reports zu unerwünschten Nebeneffekten führen.

Die erste Untermenge in Abb. 3.1 setzt sich aus allen Datensätzen zusammen, die im Feld ABT den Eintrag „TEC“ aufweisen. Das Feld ABT entscheidet über den Unterschied zwischen den Untermengen mit „TEC“ und „BHT“.

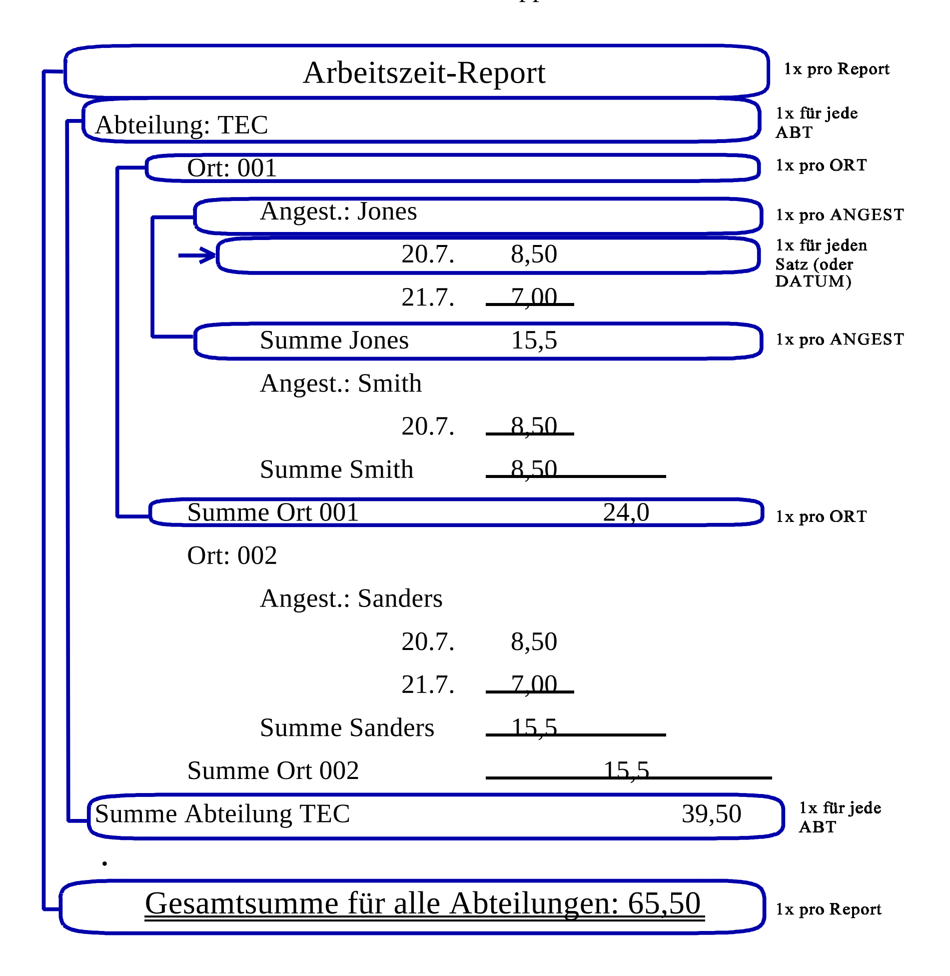


Abb. 3.2: Reportgruppen

Der Gruppenausdruck für die Gruppe würde dann **"DBF->ABT"** lauten (vorausgesetzt die Datendatei heißt DBF). Wichtig ist, daß der Gruppenausdruck nicht **"DBF->ABT='TEC'"** lautet. Das liegt daran, daß der Gruppenausdruck keine bestimmte Untermenge, sondern die Grundlage der Untermenge beschreibt. Ändert sich der ausgewertete Gruppenausdruck, gilt eine neue Untermenge, und die Funktionen der Gruppe werden ausgeführt. Die Gruppe wird aufgrund ihres Gruppenausdrucks gebildet.

Kopf und Fuß

Wie aus Abb. 3.2 ersichtlich, läßt sich eine Gruppe am Anfang und am Ende der Untermenge durch einen Textabschnitt begrenzen. Diese Abschnitte oder Reportbereiche (siehe Kapitel „Bereiche“) sind mit der Gruppe verbunden und bilden eine Einheit mit der Funktion der Gruppe.

Der Bereich am Anfang der Gruppe (z. B. „Abteilung: TEC“ in Abb. 3.2) wird als Gruppenkopf und der Bereich am Ende (z. B. „Summe Abteilung TEC“) als Gruppenfuß bezeichnet.

Der Kopf und/oder Fuß einer Gruppe können keine, einen oder mehrere Reportbereiche umfassen. Dabei handelt es sich um Abschnitte des Reports, in die Ausgabeobjekte eingefügt werden können (siehe die Kapitel „Bereiche“ und „Ausgabeobjekte“). Gruppenkopf und -fuß sind dazu da, die Funktionen der Gruppe auszuführen; das heißt, daß beim Auftreten einer Gruppenrücksetzbedingung die mit Kopf und Fuß verbundenen Reportbereiche ausgegeben werden.

Der Hauptunterschied zwischen den Reportbereichen, die mit dem Kopf bzw. dem Fuß verbunden sind, besteht darin, daß

- die Werte der im Kopfbereich ausgegebenen Objekte auf dem zusammengesetzten Datensatz basieren, der die Rücksetzbedingung verursacht hat, und
- die Werte der im Fußbereich ausgegebenen Objekte auf dem dem zusammengesetzten Datensatz unmittelbar vorausgehenden Datensatz basieren, der die Rücksetzbedingung verursacht hat.

Aus Abb. 3.3 geht dieser etwas kompliziert klingende Sachverhalt deutlich hervor. Nach dem Anlegen einer Gruppe enthält diese automatisch einen Kopf- und einen Fußbereich.

Gruppen anlegen

Eine Gruppe wird über die Menüoption **GRUPPE | NEU** angelegt. Per Voreinstellung wird jede neue Gruppe so positioniert, daß alle bereits vorhandenen Gruppen von Kopf und Fuß der neuen Gruppe umschlossen werden. Aufgrund dieser voreingestellten Operation werden die neuen Gruppen auf der höchsten (äußersten) Stufe angelegt.

Wird eine Gruppe innerhalb einer anderen Gruppe plaziert, spricht man von Verschachtelung.

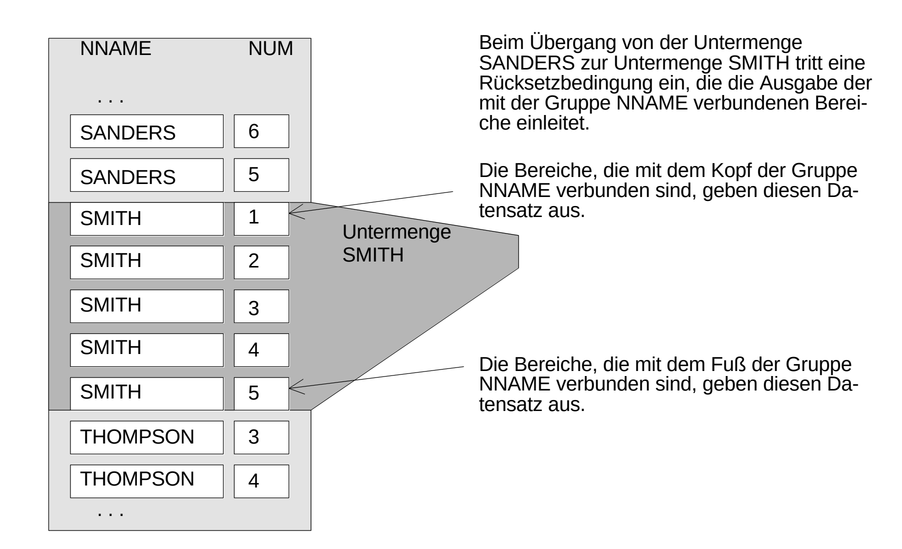


Abb. 3.3: Kopf- und Fußbereiche

Wenn eine neue Gruppe angelegt wird, erscheint gleichzeitig ein neues Informationsfenster mit Angaben zu der Gruppe sowie zu Kopf- und Fußbereich(en). Das Informationsfenster läßt sich über die Menüoption **ANSICHT | INFO-FENSTER** aus- bzw. wieder einblenden. In den Kapiteln „Starten mit CodeReporter“ und „CodeReporter-Optionen“ finden Sie Einzelheiten über das Informationsfenster.

Mit Hilfe der Menüoption **GRUPPE | NEU** rufen Sie die Dialogbox „Gruppendefinition“ auf.

Name

Eine Gruppe sollte einen sprechenden Namen haben, dem der Reportentwickler den Inhalt der Gruppe entnehmen kann. Per Voreinstellung verwendet CodeReporter „Body“ als Bezeichnung für die erste Gruppe, „Gruppe 2“ für die zweite, „Gruppe 3“ für die dritte usw.

Der Gruppenname, der lediglich ein die Gruppe beschreibendes Element darstellt, läßt sich je nach Wunsch austauschen oder verändern. Gewöhnlich schlägt sich der Gruppenausdruck irgendwie in dieser Bezeichnung nieder.

Position

Die „Position“ bestimmt, an welche Stelle die Gruppe in bezug auf andere Gruppen gesetzt wird. Die innerste Gruppe des Reports (die kleinste Untermenge) ist Gruppe 1. Per Voreinstellung werden neue Gruppen immer außen angelegt und haben den höchsten Gruppenzähler. Über die Veränderung dieses Zählers erhält die Gruppe die gewünschte Position, und die anderen Gruppen verschieben sich entsprechend.

Gruppenoptionen

CodeReporter stellt Ihnen einige besondere Funktionen zur Verfügung, mit denen Sie das Erscheinungsbild von Reports Ihren Erfordernissen anpassen können.

Kopf austauschen/Fuß austauschen

Trifft eine Gruppe auf eine Rücksetzbedingung und ist die Schaltfläche „Kopf austauschen“ für die Gruppe aktiviert, wird eine neue Seite begonnen und der Gruppenkopf statt des Seitenkopf-Bereichs ausgegeben. Auf dieser Seite wird der Seitenkopf unterdrückt.

Hinweis: Dieses Konzept ist zunächst nicht ganz leicht zu verstehen. Am besten schauen Sie sich Abb. 3.4 genau an und denken gründlich über die folgenden Ausführungen nach. Sehen Sie sich bitte ebenfalls die Übung zum „Kontoauszugsreport“ noch einmal an.

Abb. 3.4: Dialogbox „Gruppendefinition“

Ähnlich wird, wenn die Gruppe auf eine Rücksetzbedingung trifft und die Schaltfläche „Fuß austauschen“ für die Gruppe aktiviert ist, der Rest der aktuellen Seite übersprungen und der Gruppenfuß statt des Seitenfuß-Bereichs ausgegeben. Auf dieser Seite wird der Seitenfuß unterdrückt.

Im allgemeinen dient ein ausgetauschter Kopf dazu, andere Daten bzw. Daten in einem Format anzuzeigen, das von dem des Seitenkopf-Bereiches abweicht.

Der Hauptgrund für einen ausgetauschten Fuß bzw. Kopf besteht darin, den Seitenkopf/-fuß zu unterdrücken, eine neue Seite zu beginnen und den Gruppenbereich auszugeben, sobald eine Gruppenrücksetzbedingung eintritt. Die Verwendung unterscheidet sich von der für den Seitenkopf-/-fuß-Bereich verfügbaren Bereichsunterdrückungsbedingung (siehe Abschnitt „Einen Bereich unterdrükken“ im Kapitel „Bereiche“), weil der Bereich aufgrund einer Wertänderung und nicht aufgrund einer logischen Bedingung unterdrückt wird.

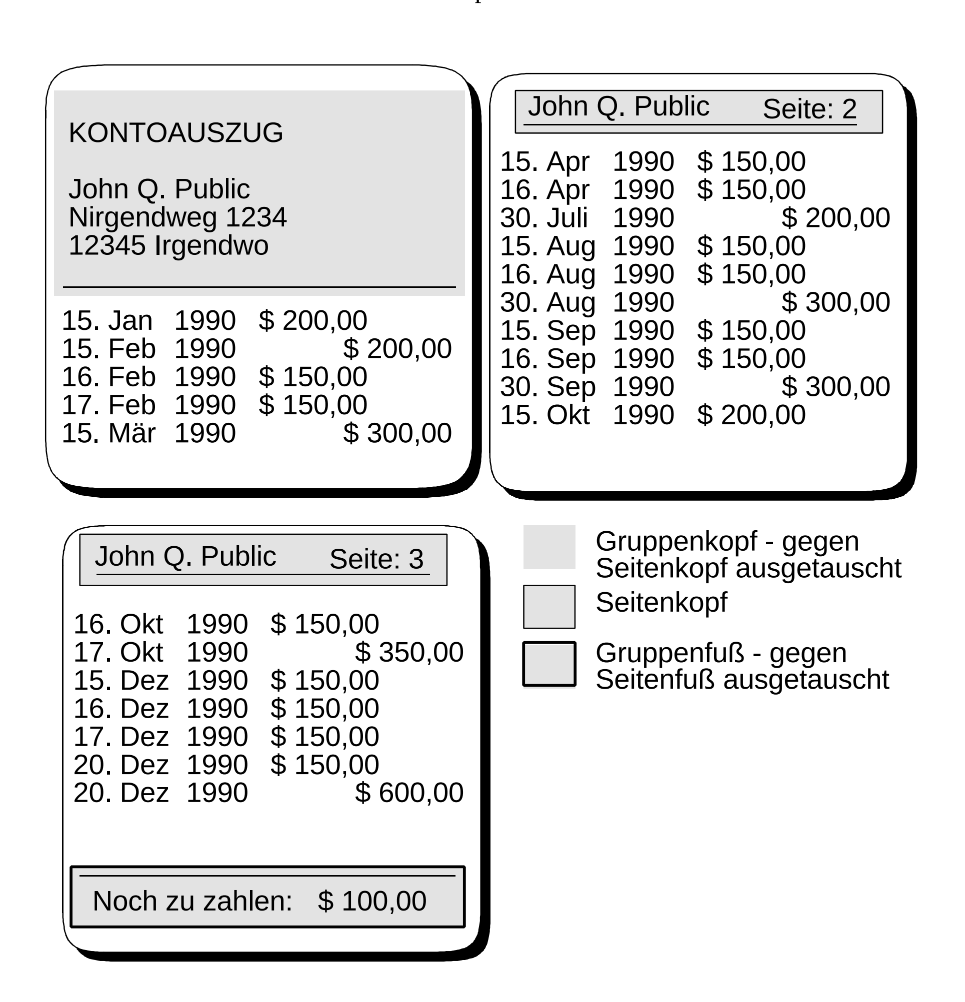


Abb. 3.5: Beispiel für Kopf/Fuß austauschen

In Abb. 3.5 sehen Sie ein Beispiel für einen Report mit ausgetauschtem Kopf und Fuß. Der auf den Seiten 2 und 3 verwendete Seitenkopf eignet sich nicht für Seite 1, da die im Seitenkopf vorhandene Zusammenfassung bereits im Gruppenkopf enthalten ist. Bei jedem Kunden wird eine neue Seite begonnen, auf der die Kundendaten in ausführlicher Form erscheinen.

Würde anstelle des ausgetauschten Gruppenkopfes ein einfacher Seitenkopf verwendet, würde die rahmenlose graue Fläche (als Seitenkopf) auf jeder Seite des Kontoauszugs gedruckt werden – so daß sich das Deckblatt und die restlichen Seiten kaum unterschieden.

In Abb. 3.5 gibt es keinen Seitenfuß-Bereich. Sind Seitenkopf/-fuß nicht vorhanden, wird an deren Stelle der Gruppenkopf/-fuß ausgegeben.

Kopf wiederholen

Reicht der Rest der aktuellen Seite nicht für eine Untermenge der zusammengesetzten Datendatei (von einer Gruppe dargestellt) aus, veranlaßt die aktivierte Schaltfläche „Kopf wiederholen“ CodeReporter dazu, auch ohne Rücksetzbedingung, den Gruppenkopf-Bereich oben auf der Seite (unterhalb des Seitenkopfes) auszugeben. Das ist eine nützliche Einrichtung, um z. B. Spaltentitel auf den Folgeseiten zu wiederholen.

Seite zurücksetzen

Die Schaltfläche „Seite zurücksetzen“ generiert beim Eintreten einer Gruppenrücksetzbedingung eine neue Seite. Dabei geschieht folgendes: der Fußbereich der Gruppe sowie die Fußbereiche aller inneren Gruppen werden ausgegeben, und eine neue Seite wird generiert. Sowohl der Kopf der nächsten Seite als auch der Gruppenkopf werden an den Anfang der neuen Seite gestellt.

Seitenzahl zurücksetzen

„Seitenzahl zurücksetzen“ ist in der Funktion identisch mit der Option „Seite zurücksetzen“, mit dem Unterschied allerdings, daß in diesem Fall die Seitenzahl der neuen Seite auf „1“ zurückgesetzt wird.

Flag „Feste Rücksetzung“

Das Verhalten der Option „Seite zurücksetzen“ (einschließlich der Zurücksetzung bei „Kopf austauschen“ und „Seitenzahl zurücksetzen“) wird von der Einstellung der Option „Flag Feste Rücksetzung“ in der Dialogbox „Reportvorgaben“ beeinflußt. Einzelheiten hierzu finden Sie im Kapitel „Reports gestalten“.

Eine Gruppe bearbeiten

Die Dialogbox „Gruppendefinition“ wird auch dazu verwendet, die Einstellungen einer bereits vorhandenen Gruppe zu verändern. Nachdem die Gruppe angelegt worden ist, können Sie die Dialogbox entweder über **GRUPPE | BEARBEITEN** oder durch Klicken der rechten Maustaste auf das Gruppeninformationsfenster aufrufen.

Einzelheiten zu den Einstellungen der Dialogbox „Gruppendefinition“ finden Sie weiter oben unter der Überschrift „Gruppen anlegen“.

Eine Gruppe löschen

Die aktuelle Gruppe sowie alle mit ihr verbundenen Bereiche und Ausgabeobjekte können Sie mit Hilfe der Menüoption **GRUPPE | LÖSCHEN** löschen. Dabei wirkt sich dieser Vorgang nicht auf andere Gruppen innerhalb des Reports aus.

Eine Gruppe auswählen

Eine Gruppe wählen Sie aus, indem Sie die linke Maustaste an einer beliebigen Stelle innerhalb des Gruppenbereichs bzw. auf ein Objekt in einem Bereich klikken.

Alternativ dazu können Sie mit den Tasten Strg_8 und Strg_9 eine neue Gruppe auswählen.

Rücksetzbedingungen und Drucken von Gruppen

Ändert sich der Gruppenausdruck, tritt eine Rücksetzbedingung ein. Diese Zurücksetzung bewirkt die Erstellung des Gruppenfußes, die Zurücksetzung der Summen und schließlich die Erstellung des Gruppenkopfes mit dem nächsten zusammengesetzten Datensatz.

Bei einem Report, in den viele Gruppen eingehen, setzt eine Rücksetzbedingung die Gruppe mit dem geänderten Ausdruck sowie alle ihr untergeordneten Gruppen (das sind die Gruppen, die sich zwischen Kopf und Fuß der ersten Gruppe befinden) zurück.

In Abb. 3.6 sehen Sie einen Report mit Gruppen für Jahr, Monat und Tag. In dem Augenblick, in dem die Gruppe „Monat" zurückgesetzt wird, wird die Gruppe „Tag" automatisch ebenfalls zurückgesetzt.

Gruppen werden lediglich bei der Rücksetzung ausgegeben. In Abb. 3.6 ist der zweite Datensatz völlig identisch mit dem ersten. Da für jede Gruppe eine Rücksetzbedingung gilt – und keine zurückgesetzt wurde –, wird der zweite Datensatz nicht ausgegeben. Mit dem Abarbeiten des dritten Datensatzes (*20. Feb 1992*) jedoch wird die Gruppe „Monat" zurückgesetzt. Automatisch werden auch alle untergeordneten Gruppen (in diesem Fall „Tag") zurückgesetzt.

Beim Eintreten einer Rücksetzbedingung werden die Fußbereiche, beginnend mit dem der innersten Gruppe, ausgegeben, bis der Fuß der Gruppe ausgegeben wird, die die Rücksetzbedingung verursacht hat.

In Abb. 3.6 wird zunächst der Fuß der Gruppe „Tag" und dann der der Gruppe „Monat" ausgegeben. Der Fuß der Gruppe „Jahr" wird zu diesem Zeitpunkt nicht ausgegeben, da diese Gruppe noch nicht zurückgesetzt wurde. In den Fußbereichen von „Tag" und „Monat" lautet der Datumswert *20. Jan 1992*, der zweite Datensatz, und nicht *20. Feb 1992*, der Datensatz der die Rücksetzbedingung verursacht hat. Hieran wird die zweite im Abschnitt „Kopf und Fuß" erwähnte Regel deutlich: Fußbereiche werden mit den Werten des Datensatzes ausgegeben, der dem Datensatz unmittelbar vorausgeht, der die Rücksetzbedingung verursacht hat.

Nachdem die innerste Gruppe verarbeitet worden ist, kommt der nächste Datensatz an die Reihe. In Abb. 3.6 setzt der Datensatz *21. Feb 1992* die Gruppe „Tag" zurück. Da lediglich diese Gruppe zurückgesetzt wird, wird auch nur der Fuß für die Gruppe „Tag" ausgegeben – wiederum mit den Werten des vorhergehenden Datensatzes; danach wird der Gruppenkopf mit den Werten des neuen Datensatzes ausgegeben.

Nachdem alle Datensätze verarbeitet worden sind, endet der Report mit der Ausgabe der Fußbereiche aller in ihm enthaltenen Gruppen. Kurzum, wird eine Gruppe zurückgesetzt,

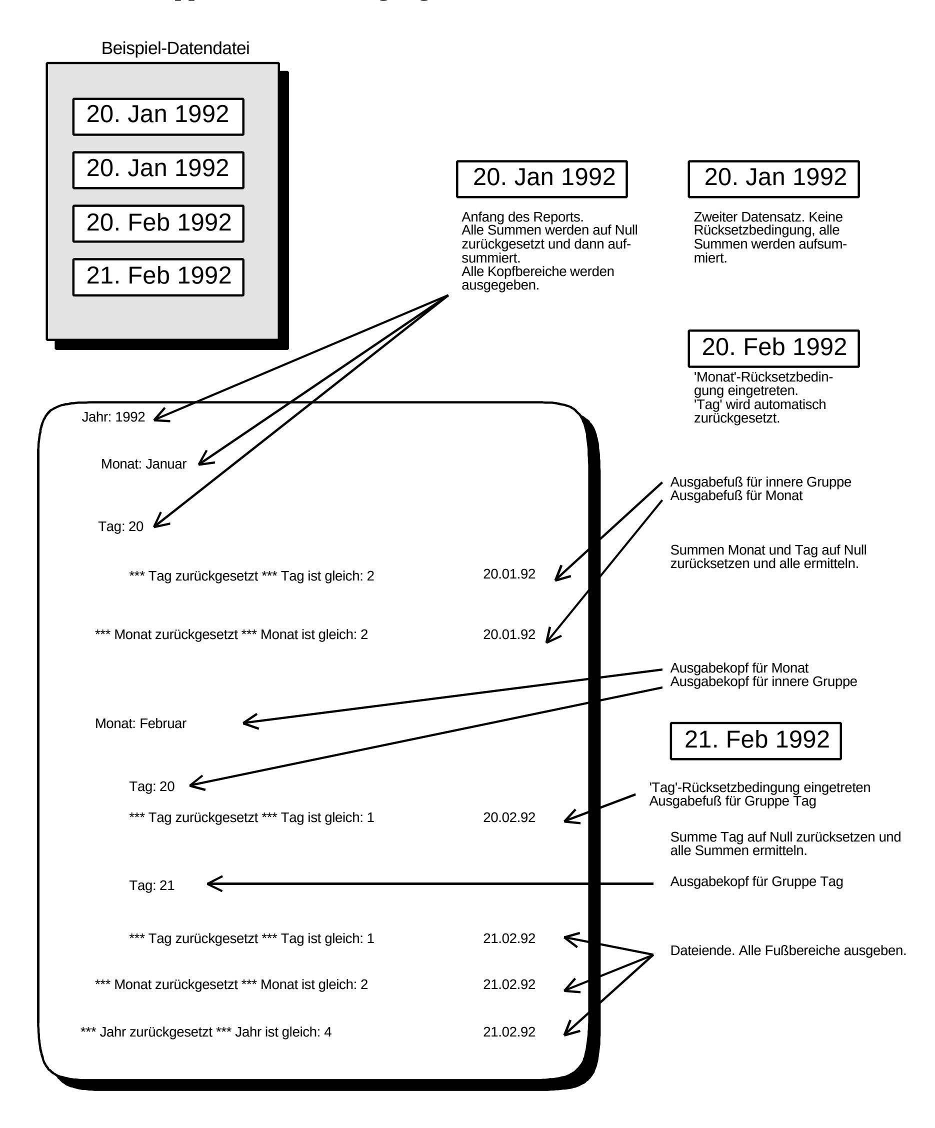
- werden vor der Ausgabe der Kopfbereiche zunächst die Fußbereiche aller Gruppen ausgegeben. Die Fußbereiche werden, beginnend mit dem der innersten Gruppe, bis hin zur Gruppe ausgegeben, die für die Rücksetzbedingung

verantwortlich ist. Der Wert des Datensatzes unmittelbar vor dem Datensatz, der die Rücksetzbedingung verursacht hat, wird bei der Ausgabe der Gruppenfüße verwendet.

- Ist bei einer der zurückgesetzten Gruppen die Option „Seite zurücksetzen“ aktiv, wird im Report eine neue Seite geöffnet. (Siehe „Seite zurücksetzen“ weiter oben sowie „Feste Rücksetzung“ im Kapitel „Reports gestalten“.)
- Nachdem die Fußbereiche verarbeitet worden sind, erfolgt die Verarbeitung der Kopfbereiche nach innen. Begonnen wird dabei mit der Gruppe, die zurückgesetzt wurde, und mit dem Wert des Datensatzes, der die Rücksetzbedingung verursacht hat.

Hinweis: Gruppen, die sich außerhalb der Gruppe mit der Rücksetzbedingung befinden, werden nicht verarbeitet. Beim vorliegenden Beispiel wird die Gruppe „Jahr“ erst zurückgesetzt, nachdem ihr eigener Gruppenausdruck sich geändert hat.

Abb. 3.6: Gruppenrücksetzbedingungen



4 Bereiche

Ein Bereich ist ein Teil eines Reports, in den Ausgabeobjekte gesetzt werden können; Seitenkopf, Seitenfuß und Hauptteil des Reports werden als Bereiche betrachtet. Von einigen Ausnahmen abgesehen (Seitenkop/-fuß, Titel/Schluß), sind alle Bereiche mit Gruppen verbunden, die die Ausgabe von Objekten innerhalb des Bereichs bestimmen. Trifft eine Gruppe auf eine Rücksetzbedingung, werden die mit der Gruppe verbundenen Bereiche ausgegeben.

Beim Anlegen einer Gruppe erhält diese standardmäßig die Bereiche Gruppenkopf und Gruppenfuß – bei der Grundgruppe „BODY“ ist lediglich ein Gruppenkopf vorhanden. Diese Bereiche können als Voreinstellung für die Gruppenbereiche des Reports übernommen oder aber je nach Bedarf in der Größe verändert, gelöscht oder unterdrückt werden.

Mit Bereichen läßt sich leicht arbeiten, und doch sind sie sehr flexibel. Neben der Anordnung von Ausgabeobjekten kann eine Reihe sich gegenseitig ausschließender unterdrückter Bereiche das Aussehen des Reports zu verändern. Eine Gruppe kann mehrere Kopf- und/oder Fußbereiche haben, die – je nach Kontext – an verschiedenen Stellen innerhalb des Reports ausgegeben werden. Darüber hinaus läßt sich ein Bereich so gestalten, daß er sich auf der folgenden Seite fortsetzt.

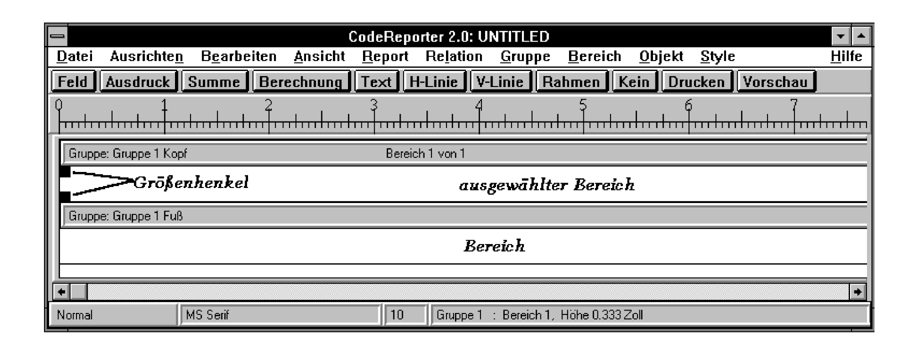


Abb. 4.1: Bereiche und Größenhenkel

Einen Bereich auswählen

Ein Bereich wird ausgewählt, indem Sie die linke Maustaste auf eine beliebige Stelle oder auf ein Objekt innerhalb des Bereichs klicken.

Alternativ dazu können Sie mit den Tasten Strg_8 und Strg_9 einen neuen Bereich auswählen. Bedienen Sie sich dieser Möglichkeit, wählen Sie damit gleichzeitig das erste Ausgabeobjekt an, das zu dem Bereich hinzugefügt wurde.

Nachdem Sie einen Bereich ausgewählt haben, können Sie ihn bearbeiten, löschen oder einen zusätzlichen Bereich für die aktuelle Gruppe definieren.

Einen Bereich anlegen

Über die Menüoption **Bereich | Neuer Kopfbereich** bzw. **Bereich | Neuer Fussbereich** können Sie für die ausgewählte Gruppe einen Kopf- oder Fußbereich erstellen. Um einen Seitenkopf-, Seitenfuß-, Titel- oder Schlußbereich anzulegen, wählen Sie die entsprechende Option im Menü **Bereich** aus. Einzelheiten zu diesen Bereichen finden Sie weiter unten.

Einen Bereich löschen

Der aktuelle Bereich kann mit Hilfe der Menüoption **Bereich löschen** entfernt werden. Dabei werden ebenfalls alle im Bereich vorhandenen Ausgabeobjekte gelöscht.

Alle Bereiche eines Gruppenkopfes bzw. -fußes können gelöscht werden; allerdings wird das Informationsfenster des letzten Gruppenbereichs weiterhin angezeigt, wenn die Menüoption **Ansicht | Info-Fenster** aktiv ist.

Einen Bereich bearbeiten

Jeder Bereich hat drei veränderbare Merkmale: Größe, Unterdrückung ja/nein und Fortsetzung nach Seitenumbruch. Diese Optionen können in der Dialogbox „Bereich bearbeiten“ (Abb. 4.2) verändert werden; das Feld wird für den ausgewählten Bereich über die Menüoption **Bereich | Bereich bearbeiten** oder über das Klicken der rechten Maustaste innerhalb des Bereichs aufgerufen.

Größe eines Bereichs bestimmen

Die Höhe eines Bereichs läßt sich mit Hilfe der Maus bzw. über die Dialogbox „Bereich bearbeiten“ (siehe Abb. 4.2) verändern. Die Breite dagegen läßt sich für einen Bereich nicht individuell einstellen. Sie hängt jeweils von der angegebenen Seitenbreite des Reports oder den eingestellten Rändern ab.

Die Vergrößerung der Bereichshöhe wirkt sich nicht auf die Position der Ausgabeobjekte innerhalb der Gruppe aus. Wird die Bereichshöhe jedoch herabgesetzt, kann es geschehen, daß einige Ausgabeobjekte gelöscht werden, wenn sie an ihrer aktuellen Position nicht mehr in den verkleinerten Bereich passen.

Die gewünschte Größe des Bereich kann manuell in das Editierfeld „Höhe“ (Dialogbox „Bereich bearbeiten“) eingegeben werden. Die angezeigte Maßeinheit wird unter **REPORT | VORGABEN** eingestellt.

Auch mit der Maus läßt sich die Höhe des ausgewählten Bereichs verändern, indem Sie einen Henkel mit der linken Maustaste anklicken und ihn auf die gewünschte Höhe ziehen (siehe Abb. 4.1).



Abb. 4.2: Dialogbox „Bereich bearbeiten“

Seitenumbruch

Ist die Option „Seitenumbruch“ für einen Bereich aktiviert (Voreinstellung), kann der Bereich auf der nächsten Seite fortgesetzt werden. Wenn innerhalb eines Bereichs ein Seitenumbruch (Ende der Seite ohne Seitenfuß) erforderlich ist, wird der Bereich auf zwei Seiten verteilt. CodeReporter bringt soviel wie möglich auf der Seite unter, ohne dabei ein Objekt zu teilen.

Bei Rahmen, Linien und Objekten mit Zeilenumbruch, die über die ganze Höhe des Bereichs gehen, ist diese Einstellung nutzlos, denn CodeReporter verteilt auf keinen Fall ein Objekt auf zwei Seiten. Auch wenn die Schaltfläche „Seitenumbruch“ aktiviert wurde, wird der ganze Bereich desungeachtet auf die folgende Seite gesetzt.

Wenn ein Ausgabeobjekt nicht auf zwei Seiten verteilt werden darf, sollten Sie die Schaltfläche „Seitenumbruch“ deaktivieren.

Einen Bereich unterdrücken

Ein logischer dBASE-Ausdruck läßt sich mit einem Bereich verbinden, um zu bestimmen, ob der Bereich ausgegeben werden soll oder nicht. Ergibt der in das Editierfeld „Unterdrückungsbedingung“ eingegebene dBASE-Ausdruck einen logisch wahren Wert, wird der Bereich (und alle Objekte darin) ignoriert.

Diese Einrichtung läßt sich dazu benutzen, das Erscheinungsbild bzw. den Inhalt des ausgegebenen Bereichs je nach Dateninhalt des Reports verschieden zu gestalten. Dazu erstellt man zwei oder mehr Bereiche im selben Teil der Gruppe (Kopf oder Fuß), weist die verschiedenen Daten den verschiedenen Bereichen zu und bestimmt, wann welche(r) Bereich(e) unterdrückt werden soll(en).

Enthält ein Feld z. B. einen negativen numerischen Wert, könnte man ihn sinnigerweise in Rot ausgeben. Die Objekte in den beiden Versionen sind, bis auf die Ausgabe der roten Schrift im Falle von Minuszahlen, identisch. Jeder Bereich erhält eine Unterdrückungsbedingung.

Schwarze Schrift (bei negativen Werten unterdrücken): `DBF->FELD < 0`
Rote Schrift (bei positiven Werten unterdrücken): `DBF->FELD >= 0`

Ein Beispiel für das Unterdrücken eines Bereichs finden Sie am Ende des vorliegenden Kapitels.

Seitenkopf- und Seitenfuß-Bereich

Die Seitenkopf-Bereiche werden oben auf jeder Seite des Reports ausgegeben und enthalten im allgemeinen Textobjekte wie die Kurzbezeichnung des Reports, ein Datum und/oder eine Seitenzahl.

Die Seitenfuß-Bereiche werden unten auf jeder Seite ausgegeben und enthalten im allgemeinen Textobjekte wie Zwischensummen, Seitensummen und Seitenzahlen.

Werden sie nicht unterdrückt oder ist bei der Gruppe die Option „Kopf austauschen“ bzw. „Fuß austauschen“ aktiviert, erscheinen diese Bereiche auf jeder Seite des Reports. Sie werden mittels der Menüoptionen **BEREICH | NEUER SEITENKOPF-BEREICH** und **BEREICH | NEUER SEITENFUSS-BEREICH** erzeugt.

Titel- und Schlußbereich

Der Titelbereich ist der erste Bereich, der auf der ersten Reportseite, nach dem Seitenkopf-Bereich, ausgegeben wird; für jeden Report wird lediglich ein Titelbereich ausgegeben. Der Titelbereich als solcher, sozusagen das Deckblatt des Reports, enthält im allgemeinen, in Gestalt des Reportnamens, eindeutige Hinweise auf seinen Inhalt.

Soll nach dem Titelbereich eine neue Seite begonnen werden, aktivieren Sie die Schaltfläche „Neue Seite nach Titel“ in der Dialogbox „Reportvorgaben“. (Siehe Abschnitt „Reportvorgaben“ im Kapitel „Reports gestalten“.)

Ein Schlußbereich wird als letzter Bereich vor dem Seitenfuß der letzten Seite des Reports ausgegeben; jeder Report enthält lediglich einen Schlußbereich. Der Schlußbereich dient der Ausgabe von Schlußbemerkungen und/oder von den gesamten Report zusammenfassenden Zahlenwerten.

Diese Bereiche werden mittels der Menüoptionen **BEREICH | NEUER TITELBEREICH** und **BEREICH | NEUER SCHLUSSBEREICH** erzeugt.

Beispiel

Bei diesem Beispiel wird die Datendatei NUMBERS.DBF angezeigt, die sowohl positive als auch negative Zahlen enthält. Durch den Einsatz von zwei Gruppen, Unterdrückungsbedingungen und einer anderen Schrift werden die negativen Zahlen der Datendatei in Rot, die positiven Zahlen in Schwarz angezeigt.

Bei der Durcharbeitung des Beispiels setzen Sie einige der Dinge ein, die Sie im vorliegenden Kapitel gelernt haben; darüber hinaus fließen hier weitere Informationen aus den Kapiteln „Ausgabeobjekte“ und „Styles“ ein (siehe dort).

Aktivieren Sie nach dem Aufruf von CodeReporter die Menüoption **DATEI | NEU**, und wählen Sie im Verzeichnis .\EXAMPLES die Datei NUMBERS.DBF an; CodeReporter erzeugt die Gruppe „Body“ mit einem Kopfbereich.

Um die negativen Zahlen auszugeben, wird ein zweiter Gruppenkopf über die Menüoption **BEREICH | NEUER KOPFBEREICH** erzeugt. Nachdem der neue Bereich angelegt (und automatisch ausgewählt) wurde, rufen Sie über die Option **BEREICH | BEREICH BEARBEITEN** die Dialogbox „Bereich bearbeiten“ auf und geben folgenden Ausdruck in das Editierfeld „Unterdrückungsbedingung“ ein:

```
NUMBERS->NUMERIC >=0
```

Ist der Wert dieses Feldes größer oder gleich null (positive Zahl), wird der zweite Bereich nicht ausgegeben. Der zweite Bereich dient ausschließlich der Ausgabe negativer Zahlen – die positiven Zahlen werden im ersten Bereich ausgegeben. Klicken Sie nach Ihrer Eingabe auf „OK“.

Per Voreinstellung gibt es für den ersten Bereich keine Unterdrückungsbedingung, so daß jeder Wert – sowohl positiv als auch negativ – der Datendatei NUMBERS.DBF angezeigt wird. Rufen Sie also zur Eingabe einer Unterdrükkungsbedingung für den ersten Bereich die Dialogbox „Bereich bearbeiten“ auf, oder klicken Sie die linke Maustaste, während der Mauszeiger im ersten Bereich steht. Geben Sie folgenden Ausdruck in das Editierfeld „Unterdrückungsbedingung“ ein:

```
NUMBERS->NUMERIC < 0
```

Ist der Wert dieses Feldes eine negative Zahl, wird der erste (für die Ausgabe von positiven Zahlen benutzte) Bereich unterdrückt.

Klicken Sie mit der Maus in der Tastenleiste auf die Schaltfläche „Feld“, und klicken Sie in der Popup-Liste auf das Feld **NUMERIC**, um es auszuwählen.

Stellen Sie den Mauszeiger in den ersten Bereich (der Zeiger verwandelt sich in einen Feldzeiger), und drücken Sie die linke Maustaste, um das Feld zu positionieren. (Einzelheiten zum Positionieren und Verschieben von Ausgabeobjekten finden Sie im Kapitel „Ausgabeobjekte“.)

Wählen Sie das Feld **NUMERIC** in der Popup-Liste noch einmal an, und positionieren Sie es im zweiten Bereich. Klicken Sie danach in der Listbox „Feldobjekte“ auf die Schaltfläche „Fertig“.

Aktivieren Sie die Menüoption **STYLE | SEITEN-LAYOUT LADEN**, und wählen Sie das Seiten-Layout TUTORIAL.CRS aus. Wenn CodeReporter fragt, ob das aktuelle Layout überschrieben werden soll, klicken Sie auf „Ja“. Nähere Einzelheiten zum Anlegen, Speichern und Laden von Seiten-Layouts entnehmen Sie bitte dem Kapitel „Styles“.

Wählen Sie das Feld **NUMERIC** im „negativen“ Bereich durch Anklicken mit der linken Maustaste an. Aktivieren Sie die Schaltfläche „Style“ in der Statusanzeige, und Doppelklicken Sie auf „rot“. Dadurch ändert sich das Layout aller angewählten Ausgabeobjekte. Da Sie das Feld **NUMERIC** im negativen Bereich ausgewählt haben, werden seine Werte in Rot ausgegeben.

Aktivieren Sie die Menüoption **DATEI | DRUCKBILD EINSEHEN**, um sich den fertigen Report anzeigen zu lassen. Es werden alle Werte in der Datendatei NUMBERS.DBF aufgelistet. Da Sie jedoch zwei Bereiche mit sich gegenseitig ausschließenden Unterdrückungsbedingungen definiert haben, werden die negativen Zahlen in Rot ausgegeben.

5 Ausgabeobjekte

Der Begriff „Ausgabeobjekte“ umfaßt die Gesamtheit aller Reportelemente, mit deren Hilfe dem Leser der Inhalt des Reports vermittelt wird, sozusagen das Wesentliche daran – Texte, Felder, Summen, Grafiken, Linien, Rahmen usw. Diese Objekte werden beim Ausdrucken mit zu Papier gebracht. Alle Ausgaben in einem Report erfolgen über Ausgabeobjekte.

Einige Ausgabeobjekte geben überall, wo sie innerhalb eines Reports eingesetzt werden, dieselben Daten aus, während andere sich mit jedem zusammengesetzten Datensatz ändern. Ausgabeobjekte, die sich nicht verändern – z. B. Linien, Beschreibungen, Firmenlogos usw. –, bezeichnen wir als statisch. Die während der Reportentwicklung für diese Elemente festgesetzten Werte und Einstellungen bleiben bei der Ausführung des Reports so bestehen.

Die Werte anderer Ausgabeobjekte – z. B. Felder und Summen – können sich bei jedem Reportlauf, ja, von einem zusammengesetzten Datensatz zum nächsten ändern. Diese laufenden Veränderungen unterworfenen Reportelemente bezeichnen wir als dynamisch. Die Werte dynamischer Ausgabeobjekte spiegeln die Daten in der/den zusammengesetzten Datendatei(en) wider, die sich jederzeit ändern können.

Die verschiedenen Arten statischer und dynamischer Ausgabeobjekte werden im Rahmen dieses Kapitels ausführlich behandelt.

Ausgabeobjekte definieren

Für jedes Objekt gilt ein bestimmter Typ, ein Feld, eine Linie, ein Text usw. Der Typ des Objekts wird bei seiner Definition festgelegt und ändert sich danach nicht mehr.

Ausgabeobjekte lassen sich jedem Reportbereich hinzufügen, z. B. dem Seitenkopf/-fuß, dem Titel/Schluß und einem Gruppenkopf/-fuß (siehe Kapitel „Gruppen“ und „Bereiche“). Sie bleiben in ihrem ursprünglichen Bereich, es sei denn, sie werden über Ausschneiden und Einfügen, wie im Abschnitt „Objekte verschieben“ weiter unten beschrieben, verlagert.

Einfügemodus

Haben Sie einen Objekttyp zur Hinzufügung zum Report ausgewählt, begibt sich CodeReporter in den Einfügemodus. Die Statusanzeige gibt das Wort „Einfügemodus:“ und dahinter den ausgewählten Objekttyp wieder. Mit dem Mauscursor, der sich in den entsprechenden Objektcursor verwandelt (siehe Anhang C), können Sie die Ausgangsposition des neuen Objekts festlegen. Ein Klick mit der linken Maustaste plaziert das ausgewählte Objekt an der gewünschten Stelle.

Die Art und Weise der Objektdefinition ist von Typ zu Typ verschieden. Bei einigen Objekte sind Grundwerte erforderlich, während andere sich voreingestellter Werte bedienen. Im folgenden beschreiben wir ganz allgemein, wie man ein Ausgabeobjekt definiert. Ausführliche Erläuterungen zu den einzelnen Objekttypen finden Sie weiter unten.

Einsatz der Tastenleiste

Über die Tastenleiste lassen sich die verschiedenen Ausgabeobjekte am einfachsten einfügen. Um CodeReporter für den gewünschten Objekttyp in den Einfügemodus zu versetzen, brauchen Sie lediglich auf die entsprechende Schaltfläche zu klicken. Die Tastenleiste enthält bis auf Grafikobjekte alle Objekttypen; Grafikobjekte können nur über das Menü eingefügt werden.

Einsatz des Menüs

Die Menüoption **OBJEKT** zeigt eine Liste aller Ausgabeobjekte an. Versetzen Sie CodeReporter über die Auswahl der entsprechenden Option in den Einfügemodus für das gewünschte Objekt.

Mehrere Objekte definieren

Bis Sie sich für einen anderen Objekttyp entscheiden, verläßt CodeReporter den aktuellen Einfügemodus nicht. Das heißt, Sie können beliebig viele Objekte vom selben Typ einfügen, ohne daß Sie die Auswahl immer wieder neu vornehmen müssen. CodeReporter verläßt den Einfügemodus, wenn Sie in der Tastenleiste auf „Kein“ klicken, die Menüoption **OBJEKT | KEIN** verwenden oder einfach die ESC-Taste betätigen.

Objekte innerhalb von Objekten

Ein Ausgabeobjekt kann so positioniert werden, daß es ein anderes kleineres Objekt vollständig einschließt. In einem solchen Fall wird das kleinere Objekt als Teil des größeren behandelt, da es diesem quasi „innewohnt".

Veränderungen an dem größeren Objekt wirken sich auch auf alle kleineren Objekte darin aus. Ein Rahmenobjekt zum Beispiel läßt sich zur Hervorhebung um mehrere Feldobjekte plazieren. Wird nun für das Rahmenobjekt ein neues Layout ausgewählt, verwenden auch die inneren Objekte das neue Layout. Beim Verschieben des übergeordneten Objekts werden alle Objekte, die es enthält, mit verschoben. Wird das Rahmenobjekt gelöscht, werden gleichzeitig alle darin enthaltenen Feldobjekte gelöscht.

Objekte auswählen

Damit ein Objekt bearbeitet, gelöscht, verschoben usw. werden kann, muß es vorher mit Hilfe eines der unten angegebenen Verfahren ausgewählt worden sein. Das aktuelle Objekt wird auf dem Report-Entwicklungsbildschirm in roter Schrift und mit Größenhenkeln markiert.

Hinweis: Wählen Sie ein Objekt aus, das weitere Objekte enthält, werden gleichzeitig alle darin enthaltenen Objekte mit ausgewählt.

Ein Objekt kann man durch einen einfachen Mausklick darauf auswählen. Zusätzlich werden dabei Gruppe und Bereich des Objekts ausgewählt.

Mittels der TAB-Taste lassen sich Ausgabeobjekte im ausgewählten Bereich markieren. Das wiederholte Betätigen der TAB-Taste läßt die Markierung alle Objekte des Bereichs durchlaufen. Mit der Tastenkombination Groß_TAB wandert die Markierung rückwärts.

Mehrfachauswahl

Es ist oft erforderlich, eine Operation (z. B. einen Layout-Wechsel) für mehrere Objekte durchzuführen. Mehrere Ausgabeobjekte lassen sich parallel auswählen und gemeinsam verändern, indem man eines von ihnen bearbeitet.

Bei der Auswahl mehrerer Objekte werden alle mit rotem Schriftzug gekennzeichnet, doch lediglich das erste ausgewählte Objekt behält seine Größenhenkel. Die Auswahl mehrerer Objekte gleichzeitig kann erfolgen beim

- Verschieben von Objekten,
- Löschen von Objekten,
- Ausschneiden, Kopieren und Einfügen von Objekten,
- bei der Auswahl von Layouts und
- bei allen Ausrichtoptionen.

Mit der Maus wählt man mehrere Ausgabeobjekte aus, indem man die Groß-Taste gedrückt hält und die gewünschten Objekte anklickt. Auf diese Weise läßt sich jedes beliebige Objekt in jedem beliebigen Bereich auswählen. Klicken Sie bei gedrückter Groß-Taste das Objekt ein zweites Mal an, wird die Auswahl zurückgenommen.

Mit der Tastatur wählt man mehrere Ausgabeobjekte aus, indem man die Strg-Taste gedrückt hält und die TAB-Taste betätigt. Über die Tastatur lassen sich Objekte in verschiedenen Bereichen nicht gleichzeitig auswählen.

Objekte löschen

Das ausgewählte bzw. die ausgewählten Ausgabeobjekte lassen sich durch Betätigen der Entf-Taste oder über die Menüoption **OBJEKT | LÖSCHEN** aus dem Report entfernen. Wird ein Ausgabeobjekt gelöscht, das andere Objekte enthält, werden diese Objekte ebenfalls gelöscht.

Achtung! Die Löschung eines Objekts läßt sich nicht rückgängig machen. Entfernen Sie aus Versehen ein Objekt, müssen Sie mit dem Objekt noch einmal ganz von vorn beginnen.

Objekte verschieben

Innerhalb seines Bereichs läßt sich ein Ausgabeobjekt mit der Maus anklicken und an eine neue Position ziehen. Auch mehrere ausgewählte Objekte lassen sich so verschieben; dazu halten Sie die Groß-Taste während des Verschiebevorgangs

gedrückt. Mit Hilfe der Editierfelder „X:“ und „Y:“ in der Dialogbox „Objektdefinition“ lassen sich Objekte genau positionieren. Einzelheiten zu dieser Dialogbox finden Sie weiter unten unter „Objekte bearbeiten“.

Hinweis: Ausgabeobjekte können nicht aus ihrem Bereich herausgezogen und positioniert werden. Die Bewegungsfreiheit mehrerer ausgewählter Objekte in zwei und mehr Reportbereichen richtet sich nach dem kleinsten Bereich.

Feineinstellung

CodeReporter ist in der Lage, Ausgabeobjekte bis auf eine Windows-Geräteeinheit genau zu positionieren. Diese extrem kleine und genaue Maßeinheit ist oft zu klein, um mit der Maus Objekte bündig zu arrangieren.

Die Menüoption **AUSRICHTEN | FEINEINSTELLUNG** ruft die Dialogbox „Koordinaten-Feineinstellung“ auf, in der sich die horizontale bzw. vertikale Feineinstellung (in der aktuellen Maßeinheit) angeben läßt. Auf diese Weise wird die Einheit bestimmt, um die die Objekte versetzt werden. Je größer dieser Einheit, desto kleiner ist die Anzahl der möglichen Koordinaten, die ein Objekt bedecken kann. Mit einer groben Feineinstellung lassen sich Objekte rascher positionieren.

Hinweis: Eine Veränderung der Feineinstellung wirkt sich nur auf neue Objekte aus, die positioniert werden, bzw. auf zu verschiebende Objekte. Ausgabeobjekte, die sich innerhalb des Reports bereits an ihrem Platz befinden, werden nicht berührt.

Bündigkeit

Mit Hilfe der Menüoption **AUSRICHTEN** lassen sich mehrere Objekte rasch untereinander ausrichten. Das erste ausgewählte Objekt bestimmt den Stand aller nachfolgend ausgewählten Objekte.

Um die Objekte anhand des ersten ausgewählten links- oder rechtsbündig auszurichten, können Sie die Menüoption **AUSRICHTEN | LINKSBÜNDIG** bzw. **AUSRICHTEN | RECHTSBÜNDIG** einsetzen. Auch Objekte in mehreren Reportbereichen lassen sich auf diese Weise ausrichten.

Die Menüoption **AUSRICHTEN | ZENTRIERT** richtet das aktuelle Objekt innerhalb des Reportbereichs mittig aus. Sind mehrere Objekte ausgewählt, bildet die Mitte des ersten ausgewählten Objekts das Zentrum, nach dem die anderen ausgerichtet werden; das erste Objekt selbst bewegt sich dabei nicht.

Über die Menüoption **AUSRICHTEN | OBEN** bzw. **AUSRICHTEN | UNTEN** werden die Objekte nach dem oberen oder unteren Rand des ausgewählten ausgerichtet. Auf diese Weise können allerdings nur im selben Reportbereich ausgewählte Objekte ausgerichtet werden.

Waagerecht/senkrecht austreiben

Drei oder mehr Objekte können mit der Menüoption **AUSRICHTEN | SENKRECHT AUSTREIBEN** bzw. **AUSRICHTEN | WAAGERECHT AUSTREIBEN** so angeordnet werden, daß der Zwischenraum zwischen allen Objekten gleich groß ist. Nachdem die beiden Endobjekte bestimmt worden sind, verschiebt CodeReporter die dazwischenliegenden Objekte der ausgewählten Option entsprechend.

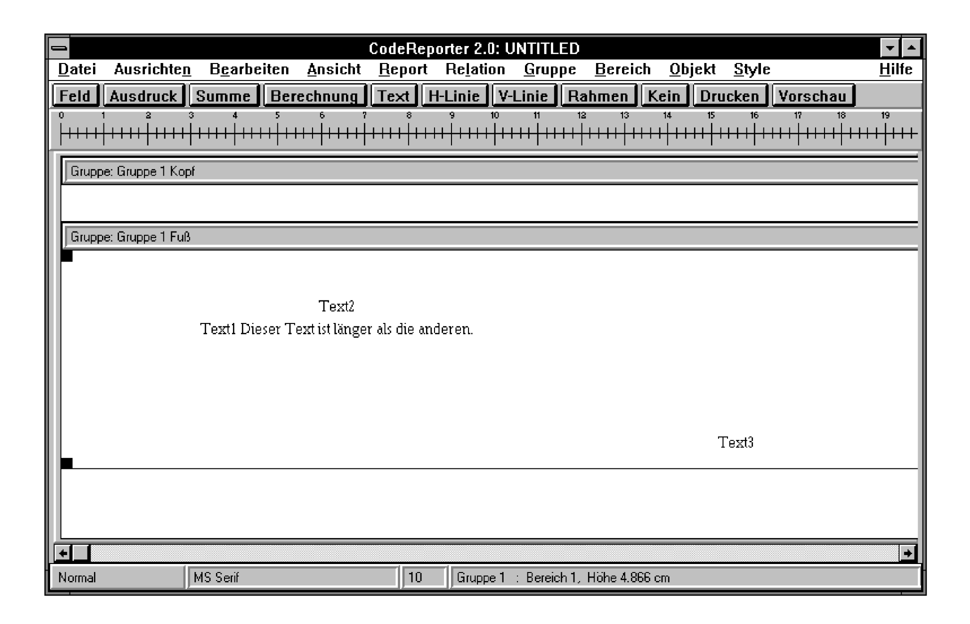


Abb. 5.1: Ausgangsformat

Abb. 5.1 enthält mehrere wahllos plazierte Ausgabeobjekte. Vorausgesetzt, sie werden in ihrer Reihenfolge markiert (also Text1, Text2 und dann Text3), werden sie über **AUSRICHTEN | LINKSBÜNDIG** so angeordnet wie in Abb. 5.2 zu sehen.

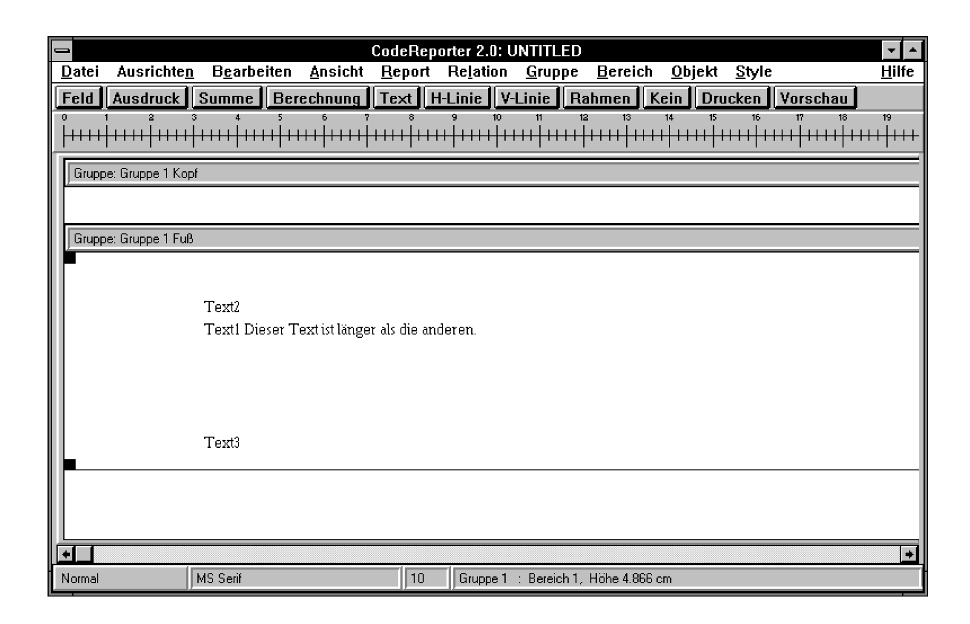


Abb. 5.2: Linksbündig ausrichten

Noch einmal von Abb. 5.1 ausgehend, werden die Ausgabeobjekte in ihrer Reihenfolge ausgewählt und mit **AUSRICHTEN | WAAGERECHT AUSTREIBEN** in die Positionen gebracht, die sie in Abb. 5.3 einnehmen. Das erste sowie das letzte ausgewählte Objekt (Text1 bzw. Text3) werden nicht versetzt, während das mittlere Objekt (Text2) gleich weit von den Enden entfernt positioniert wird.

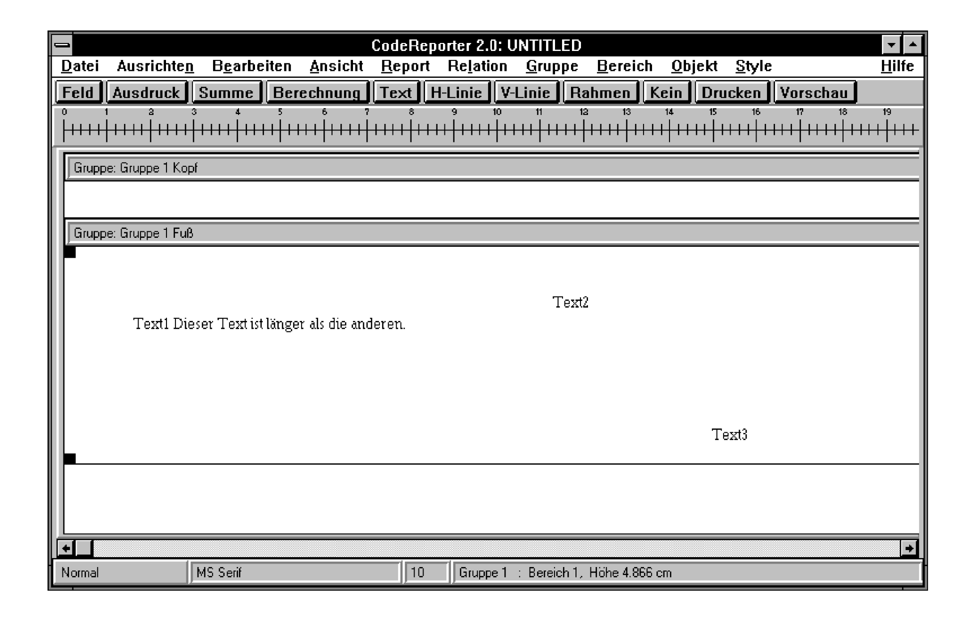


Abb. 5.3: Waagerecht austreiben

Nach vorn/hinten

Befinden sich zwei Ausgabeobjekte an derselben Stelle, verdeckt ein Objekt das andere. CodeReporter gibt anhand der Reihenfolge der Definition vor, welches Objekt über das andere gelegt wird. Das heißt, neue Objekte werden stets über bereits vorhandene positioniert.

Mittels der Menüoptionen **OBJEKT | NACH VORN** und **OBJEKT | NACH HINTEN** läßt sich die Stapelfolge für ein ausgewähltes Objekt so verändern, daß es vor bzw. hinter ein anderes Objekt gestellt werden kann.

Ausschneiden, Kopieren und Einfügen

Über die Menüoptionen **BEARBEITEN | AUSSCHNEIDEN**, **BEARBEITEN | KOPIEREN** und **BEARBEITEN | EINFÜGEN** lassen sich Objekte in andere Reportbereiche verschieben oder kopieren. Ausgeschnittene bzw. kopierte Objekte werden in der Windows-Zwischenablage gespeichert, bis sie von dort abgerufen werden.

Nach dem Aufruf der Menüoption **BEARBEITEN | EINFÜGEN** verändert sich der Mauszeiger in ein Einfügesymbol. Bewegen Sie das Symbol auf die gewünschte neue Position, und klicken Sie die linke Maustaste.

Die Zwischenablage kann dazu verwendet werden, mehrere Objekte auszuschneiden, zu kopieren und einzufügen. Ist der Zielbereich allerdings zu klein, um alle Ausgabeobjekte aufzunehmen, werden lediglich die eingefügt, die hineinpassen.

Da CodeReporter beim Ausschneiden, Kopieren und Einfügen auf die Windows-Zwischenablage zugreift, können auch die Daten anderer Anwendungen aus der Zwischenablage als Textobjekte in den Report eingefügt werden.

Grafiken aus anderen Anwendungen können als statische Grafikobjekte in einen CodeReporter-Report eingebettet werden.

Auch andere Anwendungen können Text- und statische Grafikobjekte von CodeReporter aus der Zwischenablage einlesen. Da diese Ausgabeobjekte jedoch im CodeReporter-eigenen Format vorliegen, können die anderen Anwendungen lediglich auf die Text- und statischen Grafikobjekte für die Ausgabeobjekte, nicht aber auf die Objekte selbst zugreifen.

Objekte bearbeiten

Es ist bisweilen wünschenswert, bestimmte Aspekte eines Ausgabeobjekts, wie z. B. Größe, Layout, Ausrichtung, Anzahl der Nachkommastellen usw., zu verändern. Diese Veränderungen können für das jeweilige Objekt über die Dialogbox „Objektdefinition“ vorgenommen werden, das aus dem Objektmenü des aktuellen Objekts aufgerufen wird.

Soll ein bestimmtes Ausgabeobjekt verändert werden, geschieht das über das Popup-Menü, das mit diesem Objekt verbunden ist. Je nach Objekttyp enthält das Menü drei bis vier verschiedene Auswahlpunkte. Es kann folgendermaßen aktiviert werden:

- Entweder Sie klicken mit der rechten Maustaste auf ein Objekt und halten die Taste gedrückt oder
- Sie markieren das Objekt und betätigen die Eingabetaste.

Das Objektmenü sieht danach in etwa so aus wie in Abb. 5.4 angegeben.

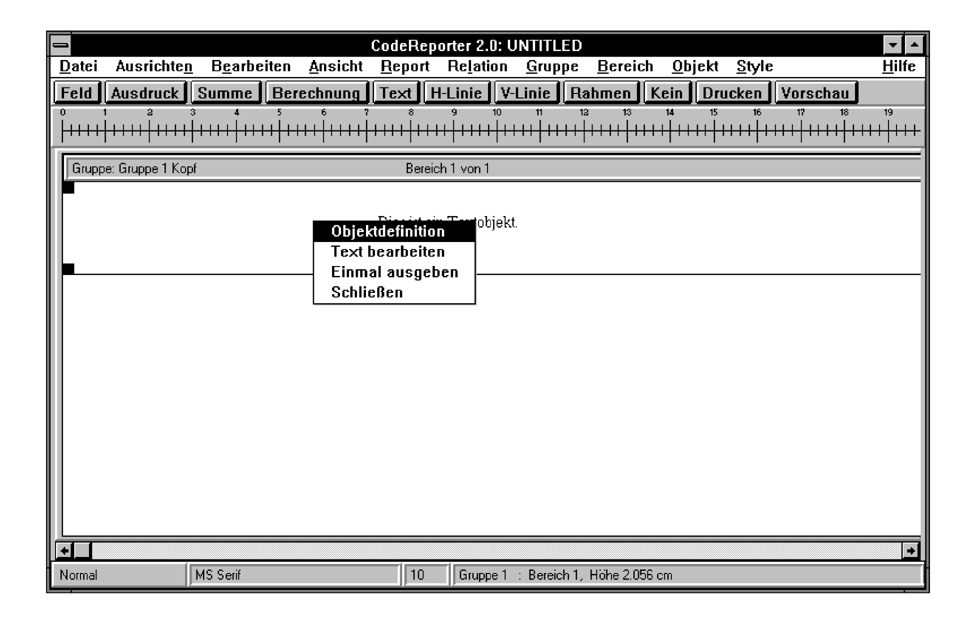


Abb. 5.4: Objektmenü

Objektdefinition

Über die Dialogbox „Objektdefinition“ lassen sich die Attribute eines Ausgabeobjekts verändern. Sie wird über die Option **OBJEKTDEFINITION** im Objektmenü aufgerufen.

Die Dialogbox selbst gliedert sich in mehrere Bereiche. Da einige Optionen nur für bestimmte Typen von Ausgabeobjekten gelten, tauchen diese Abschnitte bei anderen Ausgabeobjekten nicht auf.

Oben links enthält die Dialogbox „Objektdefinition“ eine Beschreibung des zu verändernden Objekts. Darin werden Objekttyp und Name, Ausdruck oder vom Objekt verwendeter Text angegeben.

Der Bereich „Position“ zeigt die aktuelle waagerechte („X“) und senkrechte („Y“) Position der linken oberen Ecke des Objekts an. Diese Koordinaten haben die in der Dialogbox „Reportvorgaben“ eingestellte Maßeinheit. Verändern Sie

den X- bzw. Y-Wert des Objekts, verlagert es sich auch im Entwicklungsbildschirm. Einzelheiten zum Verschieben eines Objekts mit der Maus finden Sie weiter oben.

Hinweis: Ein Objekt läßt sich nicht vollständig außerhalb seines Bereiches positionieren.

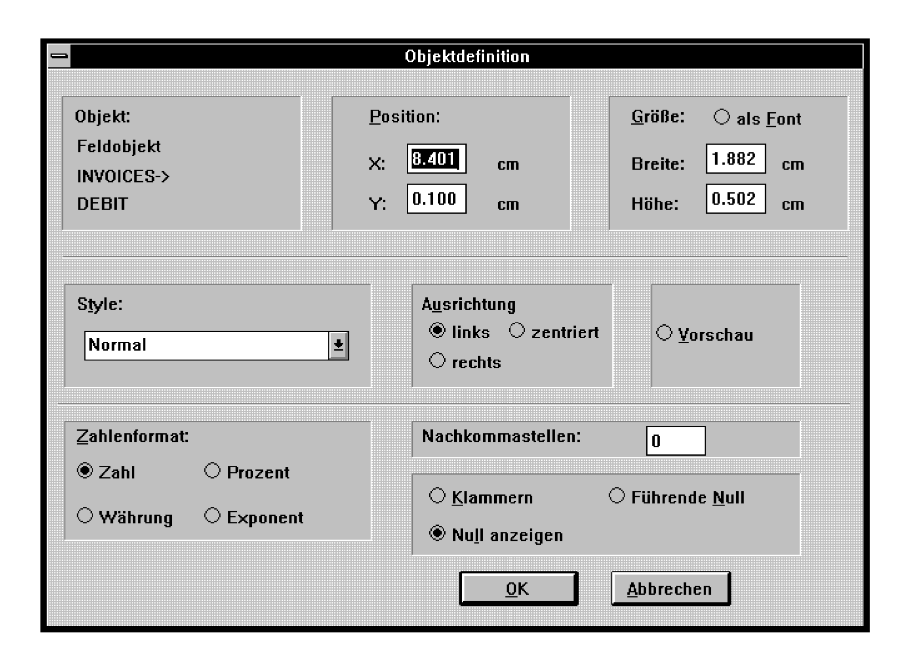


Abb. 5.5: Dialogbox „Objektdefinition“

Im Bereich „Größe“ lassen sich die aktuelle Breite und Höhe des ausgewählten Objekts ablesen. Diese Werte liegen in der in der Dialogbox „Reportvorgaben“ eingestellten Maßeinheit vor. Die Größe eines Objekts läßt sich hier exakt angeben. Wie man die Größe eines Ausgabeobjekts mit der Maus verändert, erfahren Sie weiter unten im Abschnitt „Größe verändern“.

Wird die Optionsschaltfläche „als Font“ aktiviert, werden die Werte der Editierfelder „Höhe“ und „Breite“ in die entsprechenden Zeicheneinheiten umgerechnet. Die Größe der Editierfelder läßt sich auf *n* Zeichen Breite und *m* Zeilen Höhe

einstellen. Da es sich dabei um Durchschnittswerte der Zeichenbreiten handelt, können bei Verwendung von Proportionalschriften leichte Abweichungen in der Objektgröße auftreten.

Der „Style“ (Schriftart, Farbe usw.) des Ausgabeobjekts kann über eine einzeilige Listbox verändert werden. Sie enthält lediglich bereits vorhandene Styles. Nähere Einzelheiten zu Styles und ihre Erstellung entnehmen Sie bitte dem Kapitel „Styles“.

Der Text für Ausgabeobjekte kann ausgerichtet werden. Das heißt, der für das Objekt ausgegebene Text kann links, rechts oder in der Mitte des Objekts stehen.

Links- bündig ausgerichtete s Objekt	Rechts- bündig ausgerichtete s Objekt	Mittig ausgerichtete s Objekt

Per Voreinstellung werden alle Ausgabeobjekte, bis auf numerische, die rechtsbündig stehen, linksbündig ausgerichtet. Die Ausrichtung wirkt sich auf folgende Objekttypen aus: Felder, statischer Text, Ausdrücke, Summen und Berechnungen.

Hinweis: Der Funktion TRIM() kommt bei der Ausrichtung von dBASE-Ausdrücken, die Datenbankfelder enthalten, große Bedeutung zu. Da Datenbankfelder eine feste Länge haben (fehlende Daten werden mit Leerzeichen aufgefüllt), bewirkt deren Ausrichtung wenig oder gar nichts. Ein zehn Zeichen umfassendes Ausgabeobjekt zum Beispiel enthält ungeachtet der Ausrichtung immer zehn Zeichen und ist gleichzeitig links-/rechtsbündig und mittig ausgerichtet.

Ein mittig ausgerichtetes Ausdrucksobjekt, das zehn Zeichen beinhaltet, z. B. `"SCHUH     "`, unterscheidet sich kaum von einem linksbündig ausgerichteten, da die Leerzeichen (auch bei Verwendung einer Proportionalschrift) berücksichtigt werden. Werden die Leerzeichen mit TRIM() entfernt, wird `"SCHUH     "` zu `"SCHUH"`, und die zentrierte Ausrichtung erscheint auch für das Auge korrekt.

Bei Feldausgabeobjekten werden Leerzeichen automatisch entfernt.

Proportionalschriften können auch ohne den Einsatz der Funktion TRIM() ausgerichtet werden. Allerdings wird der ausgegebene Text in diesem Fall für das Auge nicht korrekt ausgerichtet, da die Leerzeichen ungeachtet ihrer Weite in der Zeichenkette berücksichtigt werden.

Hinweis: Wir empfehlen, die Funktion TRIM() stets bei rechts- bzw. linksbündig ausgerichtetem Text einzusetzen.

Größe verändern

Die Größe eines Ausgabeobjekts läßt sich auf zweierlei Art und Weise verändern. Zum einen kann man neue Werte in das Editierfeld „Größe“ innerhalb der Dialogbox „Objektdefinition“ eingeben, zum andern die Größenhenkel des Objekts mit der Maus versetzen.

Das Editierfeld „Größe“ erlaubt die exakte Bestimmung von Höhe und Breite eines Ausgabeobjekts. Allerdings sieht man nicht, wie sich das Objekt verändert, und die Einstellung dauert länger.

Bei den Größenhenkeln handelt es sich kleine schwarze Quadrate, die den Anzeigetext eines Objekts umgeben und mit deren Hilfe die Größe des Objekts verändert werden kann (siehe Abb. 5.6). Dazu klickt man mit der linken Maustaste auf einen Henkel und zieht das Objekt mit der Maus auf die gewünschte Größe.

In Verbindung mit der Feineinstellung läßt sich mit dieser schnellen Methode die Größe eines beliebigen Ausgabeobjekts exakt einstellen. (Siehe Unterabschnitt „Feineinstellung“ weiter oben.)

Achtung! Die Größe von Grafikobjekten läßt sich über das Editierfeld „Größe“ bzw. über die Henkel verändern. Nach dem Einstellen der neuen Größe wird das Bild vergrößert und/oder verkleinert, um den neuen Raum auszufüllen. Dabei bleiben möglicherweise die ursprünglichen Proportionen nicht erhalten.



Abb. 5.6: Größenhenkel eines Objekts

Zeilenumbruch

Alle Ausgabeobjekte – bis auf Linien, Rahmen und Grafiken – geben ihren Inhalt unter Verwendung des im entsprechenden Style angegebenen Zeichensatzes aus. In den meisten Fällen ist dazu lediglich eine Zeile erforderlich. Insbesondere bei langen Feldobjekten (einschließlich Memofelder) reicht allerdings eine Zeile nicht aus.

In einem solchen Fall führt CodeReporter in dem für das Ausgabeobjekt vorgesehenen Raum Zeilenumbrüche aus. Wenn also der Text für ein Ausgabeobjekt nicht einzeilig in dem vorgesehenen Raum ausgegeben werden kann, gibt CodeReporter bis zum Rand des Objekts so viele Wörter (durch Leerstellen getrennt) wie möglich aus und setzt die restlichen – soweit in der Senkrechten Platz vorhanden ist – in die zweite Zeile des Objekts.

Hinweis: Durch Zeilenumbrüche vergrößert sich die Größe eines Objekts nicht. Kann der Text nicht innerhalb des zur Verfügung stehenden Raumes untergebracht werden, wird der Rest abgeschnitten.

Per Voreinstellung erzeugt CodeReporter einzeilige Ausgabeobjekte. Benötigen Sie ein Objekt mit mehreren Zeilen, sollten Sie dessen Größe über die Dialogbox „Objektdefinition“ bzw. die Größenhenkel verändern.

Vorschau

Die „Vorschau“-Funktion gestattet es, ein Objekt mit dem Wert im Gruppenkopf auszugeben, den es bei Ausgabe im Gruppenfuß erhalten hätte. Im wesentlichen erhält das Ausgabeobjekt auf diese Weise seinen Wert vom letzten Datensatz,

bevor eine Gruppenrücksetzbedingung eintritt. Einzelheiten zu den Unterschieden zwischen Gruppenkopf und Gruppenfuß entnehmen Sie bitte dem Abschnitt „Kopf und Fuß“ im Kapitel „Gruppen“.

Angenommen, in einem Kontenreport sollen die einzelnen Buchungsvorgänge nach Kontonummer und Datum sortiert und für jedes Konto gesondert der Zeitraum der Buchungen aufgeführt werden. Da der Report nach dem Datum sortiert wird, enthält der erste Datensatz der Gruppe (der für die Ausgabe des Gruppenkopfes verwendet wird) das Datum der Buchung auf dem Konto. Für den Anfang der Buchungszeitraums reicht ein einfaches Feldobjekt aus. Der letzte Datensatz vor der Zurücksetzung der Gruppe enthält das Datum der letzten Buchung auf dem Konto. Ein einfaches Feldobjekt, das als *Vorschauobjekt* in den Gruppenkopf gesetzt wird, gibt das Datum der letzten Buchung für das Konto aus.

Ein Vorschausummenobjekt im Gruppenkopf enthält dort denselben Wert, den es als einfaches Summenobjekt im Gruppenfuß annehmen würde.

Als praktische Begleiterscheinung kann die Aufsummierung der Gruppe vor Ausgabe der Datensätze erfolgen; die Vorschausumme läßt sich in einer internen Gruppe als Detailzeilen verwenden, die einen Prozentsatz der Gesamtsumme enthalten.

Ein nach Verkäufern sortierter Umsatzreport zum Beispiel kann den Prozentanteil eines jeden Verkäufers am Gesamtumsatz enthalten, indem in einer äußeren Gruppe einfach ein Vorschausummenobjekt und im Gruppenfuß des Verkäufers eine Berechnung angelegt wird, die die Summe der Verkäufe geteilt durch die Vorschausumme enthält.

Folgende Typen von Ausgabeobjekten lassen sich als Vorschauobjekte definieren: Felder, Summen, Berechnungen, Ausdrücke und dynamische Grafiken.

Beispiel

Als Beispiel für Vorschauausgabeobjekte erläutern wir im vorliegenden Abschnitt die zur Erstellung des oben genannten Verkaufsreports erforderlichen Schritte.

XYZ Verkaufs GmbH
Verkaufsübersicht
Gesamtverkauf: $ xx.xxx,xx

Verkäufername	Verkäufe	Prozentanteil am Gesamtverkauf
Sanders, John	$ xxx,xx	xx,xx %
Smith, John	$ xxx,xx	xx,xx %
Thompson, John	$ xxx,xx	xx,xx %

Abb. 5.7: Reportentwurf für eine Vorschau

Aktivieren Sie die Menüoption **DATEI | NEU**, und bestimmen Sie die Datei LOOKAHD.DBF zum Top Master; diese Datei finden Sie im Verzeichnis .\EXAMPLES. Der Einfachheit halber enthält sie die Namen der Verkäufer und deren Verkaufssummen.

Bei einem komplexeren Aufbau der Datendateien würde es wahrscheinlich getrennte Verkaufs- und Verkäuferdatendateien geben. In einem solchen Fall müßte eine Relation hergestellt und anstelle des Gesamtsummenfeldes im Kopf ein Summenausgabeobjekt für den Verkäufer im „Body"-Fuß verwendet werden.

Da aus der Reportbeschreibung hervorgeht, daß das Vorschausummenobjekt den gesamten Report aufsummiert, sollte es in einem Kopfbereich untergebracht werden, der nur am Anfang des Reports ausgegeben wird – z. B. dem Titelbereich. Erstellen Sie mit **BEREICH | NEUER TITELBEREICH** den Titelbereich mit einer Höhe von etwa 2,5 cm.

Plazieren Sie die beiden Felder aus LOOKAHD.DBF in den Kopfbereich von „Body". Klicken Sie dazu auf die Schaltfläche „Feld" im Entwicklungsbildschirm, wählen Sie das Feld NAME aus, bewegen Sie den Cursor in den Kopfbereich von „Body", und klicken Sie die linke Maustaste, um das Feld zu positionieren. Wiederholen Sie diese Schritte für das Feld TOTSALES. Einzelheiten zur Plazierung von Datendateifeldern in einem Report finden Sie im Abschnitt „Felder" weiter unten.

Zur Verdeutlichung sollen in den Titelbereich der Reporttitel und die Spaltentitel für die Verkäufer eingefügt werden. Klicken Sie im Entwicklungsbildschirm auf die Schaltfläche „Text“, setzen Sie den Cursor in den Titelbereich, und klicken Sie die linke Maustaste. Es erscheint die Dialogbox „Text für Textobjekt eingeben“. Geben Sie **XYZ Verkaufs GmbH** ein, und klicken Sie auf „OK“. Geben Sie anschließend in gleicher Weise die Texte „Verkaufsübersicht“, „Gesamtverkauf“, „Verkäufername“, „Verkäufe“ und „Prozentanteil am Gesamtverkauf“ ein.

Summen werden aufgrund numerischer Berechnungen und numerischer Datenfelder gebildet. Da die Verkaufssummen im Feld TOTSALES bereits vorliegen, ist keine Berechnung erforderlich.

Aktivieren Sie **OBJEKT | SUMME**, um die Dialogbox „Summenberechnungen“ aufzurufen. Wählen Sie das Feld TOTSALES aus, und setzen Sie das Summenausgabeobjekt in den Titelbereich (siehe Abb. 5.7).

Geben Sie in die Dialogbox „Summe bilden“ den Namen **TITELSUMME** und die Rücksetzbedingung für das Summenobjekt ein. Da die Summe den gesamten Report zusammenfaßt, bleibt das Editierfeld „Summenrücksetzausdruck“ leer. Klicken Sie auf „OK“. Klicken Sie in der Dialogbox „Summenberechnungen“ auf „Fertig“, um die Box zu schließen.

Alle Objekte, also auch Summenobjekte, sind von Haus aus keine Vorschauobjekte. Damit das neue Summenobjekt TITELSUMME erwartungsgemäß funktioniert, muß seine Objektdefinition verändert werden. Rufen Sie also das Objektmenü auf, und aktivieren Sie **OBJEKTDEFINITION**.

Es erscheint eine Dialogbox wie in Abb. 5.5. Klicken Sie auf die Schaltfläche „Vorschau“. Entsprechend dem Reportentwurf in Abb. 5.7 soll die Verkaufssumme mit zwei Nachkommastellen angezeigt werden. Schreiben Sie „2“ in das Editierfeld „Nachkommastellen“, und klicken Sie auf „OK“, um die Änderungen zu speichern.

Im Reportentwurf werden das Verkaufspersonal, die Verkaufssummen und der Prozentanteil am Gesamtverkauf aufgeführt. Der Prozentanteil wird anhand der Division der einzelnen Verkaufssummen durch den Gesamtverkauf ermittelt. Da der Gesamtverkauf im Titelbereich berechnet worden ist, kann er auch auf den niedrigeren Stufen des Reports verwendet werden.

Um den Prozentanteil zu ermitteln, ist ein Berechnungsobjekt erforderlich. Klikken Sie in der Tastenleiste auf die Schaltfläche „Berechnung“, um die Dialogbox

„Berechnungsobjekt“ aufzurufen. Klicken Sie auf „Neue Ber.“. Geben Sie als Namen für die Berechnung **PROZENTANTEIL** und als Berechnungsausdruck **LOOKAHD->TOTSALES/TITELSUMME()** ein, und klicken Sie auf „OK“.

Die neue Berechnung **PROZENTANTEIL** wird in die „Berechnungsobjekt“-Liste übernommen. Klicken Sie sie mit der linken Maustaste an, und plazieren Sie die Berechnung in der Gruppe „Body“. Klicken Sie auf die Schaltfläche „Fertig“, um die Dialogbox „Berechnungsobjekt“ zu schließen.

Markieren Sie das neu erstellte Berechnungsausgabeobjekt, rufen Sie das Objektmenü auf (rechte Maus- bzw. Eingabetaste), und aktivieren Sie die Menüoption „Objektdefinition“. Da der Prozentanteil in Abb. 5.7 zwei Nachkommastellen aufweist und ein Prozentwert ist, müssen diese Attribute entsprechend eingestellt werden.

Klicken Sie auf die Schaltfläche „Prozent“ im Bereich „Zahlenformat“, und ändern Sie die Anzahl der Nachkommastellen auf zwei. Klicken Sie auf „OK“, um die Dialogbox zu schließen.

VERKAUF

Nächste Schließen

XYZ Verkaufs GmbH
Verkaufsübersicht
Gesamtverkauf: $8,802.65

Verkäufername	Verkäufe	Prozentanteil am Gesamtverkauf
Smith, John	$432.10	4.91%
James, Donald	$5,092.33	57.85%
Mann, Horace	$15.00	0.17%
Levy, Eugene	$2,010.31	22.84%
Stein, Gottlieb	$430.67	4.89%
Wayne, Jonathan	$801.25	9.10%
Cypher, Scott	$20.99	0.24%

Abb. 5.8: Ausgabe mit Vorschau

Soll das Ausgabeobjekt LOOKAHD->TOTSALES als Währungswert angezeigt werden, müssen die Einstellungen ebenfalls abgeändert werden. Rufen Sie die

Dialogbox „Objektdefinition“ auf, und klicken Sie auf die Schaltfläche „Währung“ im Bereich „Zahlenformat“.

Die Entwicklung des Reports ist nun abgeschlossen. Lassen Sie sich mit **DATEI | DRUCKBILD EINSEHEN** den fertigen Report anzeigen. Sein Inhalt sollte in etwa aussehen wie der Reportentwurf und Abb. 5.8.

Zahlen

Wenn ein Ausgabeobjekt einen Zahlenwert annimmt – sei es ein numerisches Feldobjekt, eine arithmetische Berechnung, eine Summe oder ein numerischer dBASE-Ausdruck – kann die Objektausgabe formatiert erfolgen.

Zahlenformate

CodeReporter kann Zahlenwerte in vier verschiedenen Formaten ausgeben, und zwar als

- einfache Zahl (ohne Formatierung),
- Währungswert,
- Prozentwert und
- Exponent.

Bei jedem dieser Formate wird derselbe Wert unterschiedlich angezeigt.

Falls erforderlich, werden alle Zahlenwerte mit Zeichen nach Tausend und/oder Dezimalkomma ausgegeben. Diese Zeichen werden vom Report gesteuert und in der Dialogbox „Reportvorgaben“ definiert. Siehe „Reportvorgaben“ im Kapitel „Reports gestalten“, wie man diese Werte verändert.

Ein Zahlenwert kann als Währungswert formatiert werden; dazu aktivieren Sie in der Dialogbox „Objektdefinition“ die Schaltfläche „Währung“. Von sich aus setzt CodeReporter das Währungssymbol vor den eigentlichen Zahlenwert (Voreinstellung: „$“). Das Währungssymbol wird vom Report gesteuert und in der Dialogbox „Reportvorgaben“ definiert. Siehe „Reportvorgaben“ im Kapitel „Reports gestalten“.

Eine als Prozentwert formatierte Zahl wird mit hundert multipliziert, und das Prozentzeichen („%“) wird unmittelbar hinter der Zahl ergänzt.

Eine Zahl kann wissenschaftlich geschrieben und im Format *n.nnnnn e xx* ausgegeben werden; dabei ist *n* der Zahlenwert und *x* der Exponent. Bei diesem Format muß unbedingt die richtige Anzahl von Dezimalstellen angegeben werden (siehe unten).

Negative Zahl

Per Voreinstellung zeigt CodeReporter negative Zahlenwerte mit einem vorangestellten Minuszeichen (-) an. Bei bestimmten Reports müssen negative Zahlen eventuell in Klammern ausgegeben werden. Diese Einstellung erfolgt über die Schaltfläche „Klammern“ in der Dialogbox „Objektdefinition“.

Ohne Klammern	Mit Klammern
-1234,56	(1234,56)

Führende Null

Gebrochene Zahlen (zwischen 1 und -1) werden durch das Dezimalzeichen und die Nachkommastellen dargestellt. Bei einigen Objekten ist es möglicherweise wünschenswert, daß diesen Zahlen eine Null vorausgeht. Aktivieren Sie dazu die Schaltfläche „Führende Null“ in der Dialogbox „Objektdefinition“.

Führende Null	Keine führende Null
0,3	,3
33,3	33,3
-0,3	-,3

Tabelle 5.1: Führende Null

Null anzeigen

Per Voreinstellung zeigt CodeReporter numerische Ausgabeobjekte mit dem Wert Null an. Deaktivieren Sie jedoch die Schaltfläche „Null anzeigen“ in der Dialogbox „Objektdefinition“, werden numerische Ausgabeobjekte mit dem Wert Null im Report nicht ausgegeben.

Dezimalstellen

Die Anzahl der Dezimalstellen bei numerischen Objekten wird über das Editierfeld „Nachkommastellen“ in der Dialogbox „Objektdefinition“ gesteuert. Voreingestellt sind keine Dezimalstellen (bei Objekten mit numerischen Feldern allerdings werden die Dezimalstellen des Feldes verwendet).

Hat die auszugebende Zahl eine größere Genauigkeit (mehr Dezimalstellen) als definiert, wird die ausgegebene Zahl gerundet.

Achtung! Rundungen wirken sich bei Summen nicht auf den tatsächlichen Zahlenwert aus. Bei Rundungen kann es vorkommen, daß die Summe einer Zahlenspalte nicht zum Summenausgabeobjekt dieser Spalte aufsummiert wird.

Datum

In einer Datenbank werden Datumsfelder so gespeichert, daß sie sich leicht sortieren lassen. Der 2. Januar 1992 steht als „19920102“ in der Datenbank, der 3. Januar 1992 als „19920103“ usw. In gedruckten Reports läßt sich jedoch „2. Januar 1992“ bzw. „2. 1. 1992“ leichter verstehen als „19920102“.

Mit CodeReporter können Sie flexibel bestimmen, welche Formate bei der Ausgabe von Datumsobjekten verwendet werden sollen.

Datumsschablonen

Das Ausgabeformat eines Datumswertes wird mit Hilfe einer Datumsschablone dargestellt. Diese Zeichenkette enthält mehrere Formatierzeichen und/oder auch andere Zeichen. Die Formatierzeichen sind:

- **C** Jahrhundert (century). Ein „C“ steht für die erste Ziffer des Jahrhunderts. Erscheinen zwei „C“s, werden beide Ziffern des Jahrhunderts dargestellt. Alle weiteren „C“s werden nicht als Formatierzeichen benutzt.
- **Y** Jahr (year). Ein „Y“ steht für die erste Ziffer des Jahres. Erscheinen zwei „Y“s, werden beide Ziffern des Jahres dargestellt. Alle weiteren „Y“s werden nicht als Formatierzeichen benutzt.
- **M** Monat (month). Ein bzw. zwei „M“s stellen die erste oder beide Ziffern des Monats dar. Stehen mehr als drei „M“s hintereinander, wird der Monat in Buchstaben angegeben.

- **D** Tag (day). Ein bzw. zwei „D“s stellen die erste oder beide Ziffern des Monatstages dar. Alle weiteren „D“s werden nicht als Formatierzeichen benutzt.
- **Andere Zeichen**. Alle oben nicht erwähnten Zeichen werden bei der Formatierung in den Datumsstring übernommen.

Der 7. Oktober 1992 kann mit verschiedenen Datumsschablonen z. B. folgendermaßen ausgegeben werden:

FORMAT	*AUSGABE*
DD. MMMMMMM CCYY	07. Oktober 1992
YY/MMMMMMM/DD	92/Oktober/07
DD.MM.YY	07.10.92

Voreingestelltes Datumsformat

Bei allen Ausgabeobjekten mit Datum verwendet CodeReporter das voreingestellte Format DD.MM.YY. Dieses Format läßt sich über das Editierfeld „Standard-Datumsformat“ in der Dialogbox „Reportvorgaben“ abändern. Darüber hinaus kann man das voreingestellte Format objektbezogen über das Editierfeld „Datumsformat“ modifizieren (in der Dialogbox „Objektdefinition“ – in Abb. 5.5 nicht enthalten).

Das voreingestellte Datumsformat wird beim Erzeugen eines Datumsausgabeobjekts automatisch verwendet. Das verwendete Format behält seine Gültigkeit, solange das Objekt existiert (es sei denn, das Editierfeld „Datumsformat“ wird geändert). Auch eine spätere Veränderung des Datumsformats für den Report berührt das Objekt nicht.

Wenn Sie das Datumsformat eines Objekts löschen und auf „OK“ klicken, wird das Standard-Datumsformat des Reports wieder in Kraft gesetzt.

Einmal ausgeben

In einigen Fällen ist es wünschenswert, ein Objekt nur bei einer Wertänderung und nicht bei jeder Gruppenrücksetzung auszugeben. Abb. 5.9 zum Beispiel zeigt einen einfachen Report, der den Inhalt einer Datendatei ausgibt. Anstatt den Monat in jeder Zeile auszugeben, wird er nur angezeigt, wenn er sich ändert.

Diese selektive Ausgabe läßt sich über die Option **EINMAL AUSGEBEN** im Objektmenü steuern, die die Dialogbox „Objektausgabe unterdrücken“ aufruft. Klicken Sie das Kontrollkästchen „Einmal ausgeben“ an, und geben Sie einen Ausdruck zur Unterdrückung der Objektausgabe ein. Dieser Ausdruck wird bei der Gruppenrücksetzbedingung ausgewertet; weicht der ermittelte Wert von der letzten Rücksetzbedingung ab, erfolgt die Ausgabe.

Hinweis: Wie man die Ausgabe mit Hilfe einer Wahr-/Falsch-Bedingung steuert, entnehmen Sie bitte dem Abschnitt „Einen Bereich unterdrücken“ im Kapitel „Bereiche“.

Das Ausgabeobjekt **DBF->MONAT** in Abb. 5.9 verwendet die Option „Einmal ausgeben“ mit einem Unterdrückungsausdruck, der eben diesen Ausgabewert enthält. Mit der ersten Ausgabe von „Januar“ wird der Monat ausgegeben. Da sich beim zweiten Eintrag der Wert der Einmal-ausgeben-Bedingung von **DBF->MONAT** nicht ändert, wird der Monat nicht ausgegeben.

Um die Einmal-anzeigen-Bedingung für ein Objekt zu ändern bzw. zu stornieren, aktivieren Sie **EINMAL AUSGEBEN** im Objektmenü. Nehmen Sie nach dem Erscheinen der Dialogbox „Objektausgabe unterdrücken“ die erforderlichen Änderungen an der Unterdrückungsbedingung vor, oder deaktivieren Sie das Kontrollkästchen „Einmal ausgeben“, damit das Ausgabeobjekt bei jeder Zurücksetzung der Gruppe angezeigt wird.

Sie können alle Ausgabeobjekttypen auf „Einmal ausgeben“ einrichten.

Textobjekte

Ein Ausgabeobjekt einfachster Art ist ein Objekt mit statischem Text. Ein solches Objekt besteht aus einer alphanumerischen Zeichenkette, die wörtlich im Report wiedergegeben wird.

Oft dienen Textobjekte zur Bezeichnung des Reports (z. B. Titel), zur Erläuterung von nicht eindeutigen Reportelementen oder dazu, bestimmte Teile des Reports hervorzuheben.

Wenn Sie die Menüoption **OBJEKT | TEXT** anwählen oder auf die Schaltfläche „Text“ klicken, begibt sich CodeReporter in den Einfügemodus. Nach der Plazierung eines Textobjekts wird die Dialogbox „Text für Textobjekt eingeben“ aufgerufen.

Darüber hinaus werden Textobjekte erzeugt, wenn über die Menüoption **BEARBEITEN | EINFÜGEN** Texte aus der Windows-Zwischenablage (z. B. aus einer Textverarbeitung) in den Report eingelesen werden.

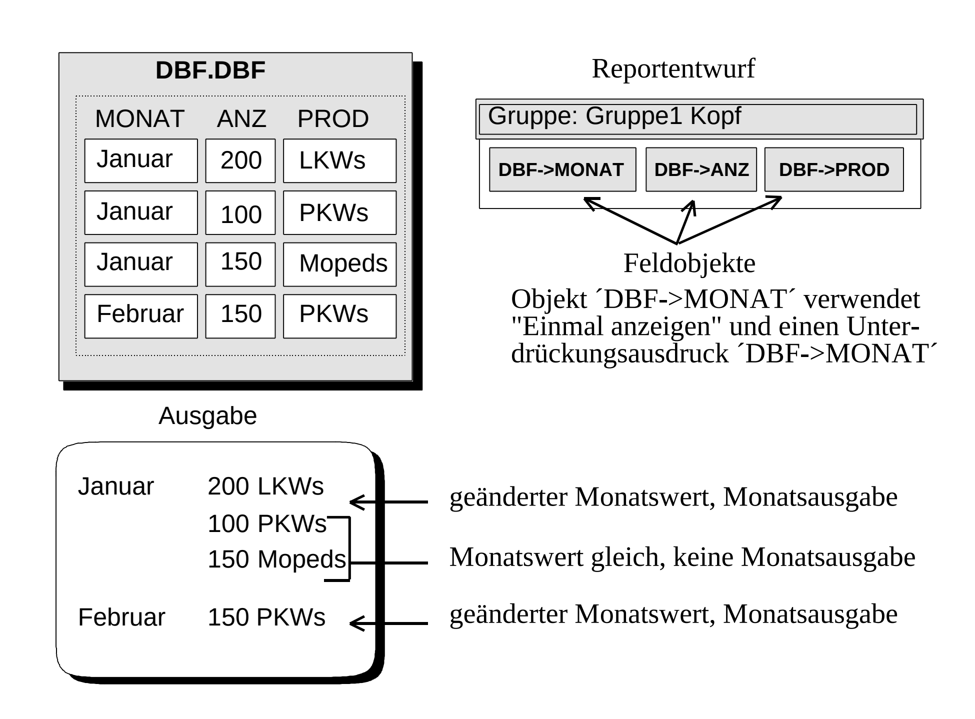


Abb. 5.9: Beispiel für „Einmal ausgeben“

Linien und Rahmen

Eine weitere Möglichkeit, einen Teil des Reports hervorzuheben, besteht in der Verwendung statischer Linien und Rahmen. Die Linien können waagerecht oder senkrecht verlaufen. Bei den Rahmen handelt es sich einfach um Rechtecke, eventuell mit Füllung oder runden Ecken. Abb. 5.10 enthält die verschiedenen Möglichkeiten bei Linien und Rahmen.

Linien

Die Erstellung einer waagerechten Linie erfolgt über einen Klick auf die Schaltfläche „H-Linie“ (oder die Menüoption **OBJEKT | WAAGERECHTE LINIE**) und einen weiteren Klick auf die Stelle, an der die Linie erscheinen soll. In gleicher Weise wird eine senkrechte Linie über die Schaltfläche „V-Linie“ (oder die Menüoption **OBJEKT | SENKRECHTE LINIE**) erstellt.

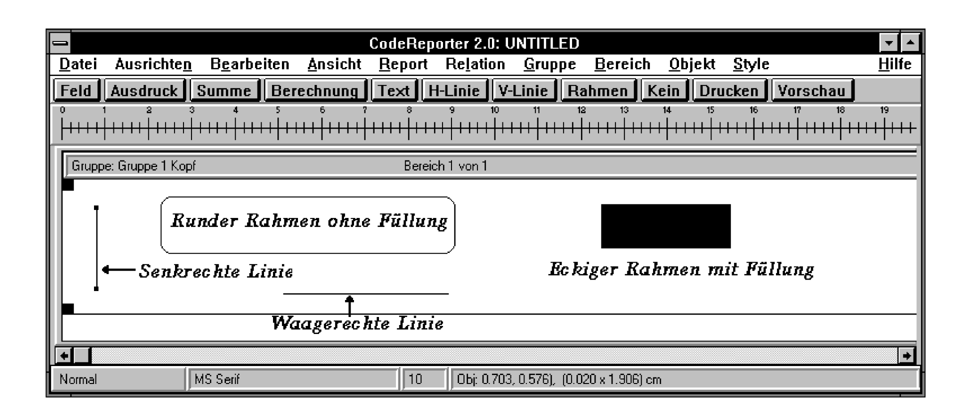


Abb. 5.10: Linien und Rahmen

Bei den Linien sind Länge und Stärke voreingestellt; diese Werte lassen sich nach der Auswahl der Option **OBJEKTDEFINITION** im Objektmenü in der Dialogbox

„Objektdefinition“ verändern. In diesem Dialog kann die Linienstärke vergrößert und die Linienfarbe neu eingestellt werden (siehe unten).

Die Linienstärke wird im Editierfeld „Stärke“ angegeben. Als Einheit werden dabei Pixel verwendet, der kleinste für einen Monitor verfügbare Wert. Die Länge einer Linie läßt sich über ihre Größenhenkel oder die Eingabe eines neuen Wertes in das Editierfeld „Länge“ verändern.

Rahmen

Bei Rahmen handelt es sich um besondere Linienobjekte, mit denen sich bestimmte Informationen in einem Report herausstellen lassen.

Ein Rahmenobjekt erstellen Sie über einen Klick auf die Schaltfläche „Rahmen“ in der Tastenleiste oder durch Auswahl der Menüoption **OBJEKT | RAHMEN**. Ein Rahmen hat jeweils eine voreingestellte Breite und Höhe, gewinkelte Ecken und keine Füllung. Mit Hilfe der Henkel können Sie Breite und Höhe des Rahmens verändern.

Die Linienstärke des Rahmens wird im Editierfeld „Linienstärke“ innerhalb der Dialogbox „Objektdefinition“ angegeben.

Per Voreinstellung sind die Ecken von Rahmen gewinkelt. Sollen sie abgerundet sein, klicken Sie auf die Schaltfläche „Runde Ecken“ in der Dialogbox „Objektdefinition“.

Ein gefüllter Rahmen läßt sich mit einer dicken Linie vergleichen. Das Innere des Rechtecks hat die Farbe des bei seiner Erstellung aktiven Styles. Der Unterschied zwischen einer dicken Linie und einem gefüllten Rahmen besteht darin, daß der Rahmen mit den Größenhenkeln in der Höhe und der Breite verstellbar ist, während bei einer Linie nur eine Richtung in Frage kommt.

Durch Klicken auf die Schaltfläche „Füllung“ in der Dialogbox „Objektdefinition“ läßt sich ein Rahmen füllen. Die Deaktivierung der Schaltflächen „Füllung“ und „Runde Ecken“ setzt den Rahmen auf die Vorgabewerte (leer, gewinkelte Ecken) zurück.

Farbe

Wie andere Ausgabeobjekte können auch Linien und Rahmen farbig dargestellt werden; dazu wählt man den Style aus, der die gewünschte Farbe enthält. Alle anderen Attribute des Styles, wie z. B. Schriftart, Punktgröße, werden ignoriert.

Einzelheiten zum Anlegen und Bearbeiten von Styles entnehmen Sie bitte dem entsprechenden Kapitel.

Objekte innerhalb

Rahmen (und dicke Linien) werden oft dazu verwendet, Ausgabeobjekte zu gruppieren. Wie im Abschnitt „Objekte innerhalb von Objekten“ weiter oben erwähnt, spricht man davon, daß das zweite Objekt im ersten enthalten ist, wenn ein Objekt ein anderes vollständig einschließt.

Das ist bei Rahmen häufig der Fall, da sie andere Ausgabeobjekte umgeben. Bei der Verwendung von Rahmen mit Füllung sollten zwei Dinge beachtet werden:

1. Das weiße Rechteck, das ein Ausgabeobjekt innerhalb des Rahmens umgibt (im Entwicklungsmodus), wird bei der Reportausgabe nicht mit ausgegeben. Es zeigt lediglich die Kontur und die Größenhenkel der inneren Ausgabeobjekte an.
2. Ausgabeobjekte innerhalb eines Rahmens mit Füllung sollten eine andere Farbe haben als die Füllung. Ein schwarzer Text auf einem schwarzem Hintergrund zum Beispiel wird „unsichtbar“. Das geschieht oft, wenn ein Style für den Rahmen über das Style-Popup-Menü ausgewählt wird. Da bei der Markierung des Rahmen gleichzeitig auch alle inneren Objekte markiert werden, wird für sie derselbe Style eingestellt wie für den Rahmen.

Grafiken

CodeReporter kann sowohl statische als auch dynamische Grafikelemente in einen Report einbetten. Vom Firmenlogo bis hin zu Bildern des Mitarbeiterstabes ist alles möglich.

Aus gewöhnlichen (langweiligen) Reports lassen sich durch Einfügen von grafischen Elementen optisch ansprechende Dokumente machen. So könnte zum Beispiel in der letzten Zeile eines Finanzreports, bei Verwendung der bedingten Bereichsunterdrückung und verschiedenen Grafikelementen, je nach Ergebnis ein „Daumen-hoch“- bzw. ein „Daumen-runter“-Symbol ausgegeben werden.

Zur Zeit unterstützt CodeReporter Windows-Bitmap-Grafiken auf dreierlei Art und Weise:

- statisch durch Einbetten der Bitmap in den Report,
- statisch durch Verweis auf einen Dateinamen und
- dynamisch durch Verwendung eines Datenfeldes, das einen Dateinamen enthält.

Hinweis: Die Ausgabe kleinerer Bitmaps ist in der Regel qualitativ besser als die größerer. Aufgrund der bei der Verkleinerung hochauflösender Bitmap-Bilder auftretenden Skalierung (z. B. bei gescannten Bildern) empfehlen wir die Verwendung von Bildern mit niedriger Auflösung.

Ein Grafikobjekt erstellen

Grafikobjekte werden über die Menüoption **OBJEKT | GRAFIK** oder das Einfügen eines Grafikelements aus der Zwischenablage erstellt. Mit der ersten Methode werden Grafikobjekte erzeugt, die kaum Plattenplatz verbrauchen. Das mit der zweiten Methode erzeugte statische Grafikobjekt wird in einer von CodeReporter angelegten Bitmap-Datei gespeichert.

Andere Windows-Anwendungen, z. B. Paintbrush, erlauben dem Benutzer das Erstellen eigener Bitmap-Grafiken. Nach der Fertigstellung kann das Bild (üblicherweise mit dem Befehl „Ausschneiden“ bzw. „Kopieren“) in die Windows-Zwischenablage gestellt werden, von wo es von CodeReporter abgerufen werden kann. Das Bild kann dann über die Menüoption **BEARBEITEN | EINFÜGEN** als statisches Grafikobjekt in den Report eingebettet werden.

Die Bitmaps eingefügter Grafikobjekte werden im Reportentwicklungsbildschirm angezeigt.

Hinweis: In einen Report eingefügte Bitmaps werden in einer externen Bitmap-Datei abgelegt. Da diese Datei im allgemeinen ohne Zutun des Benutzers angelegt wird, erhält sie einen eindeutigen – und gewöhnlich nicht verständlichen – Namen. Wurde die eingefügte Bitmap bereits von einer Anwendung in einer Datei gespeichert, wird durch die Verdoppelung unnötig Plattenplatz verbraucht.

Im allgemeinen ist es günstiger, die Bitmap mit der Ausgangsanwendung als Bitmap-Datei (.BMP) zu speichern und über die Menüoption **OBJEKT | GRAFIK** ein Grafikausgabeobjekt zu erzeugen, das auf die Datei zugreift.

Die Menüoption **OBJEKT | GRAFIK** ruft die Dialogbox „Typ des Grafikobjekts angeben“ auf, in der Sie den entsprechenden Objekttyp angeben können.

Aktivieren Sie das Kontrollkästchen „Statische Grafik“, bittet CodeReporter Sie um die Eingabe des Namens der Bitmap-Datei und zeigt die Abbildung im CodeReporter-Entwicklungsbildschirm an. In der Reportdatei wird lediglich der Name der Bitmap-Datei abgelegt, aber nicht die Bitmap selbst. Auf diese Weise kann ein Reportentwickler ein Grafikobjekt einfach durch die Veränderung des Dateinamens der Bitmap-Datei gegen ein anderes austauschen.

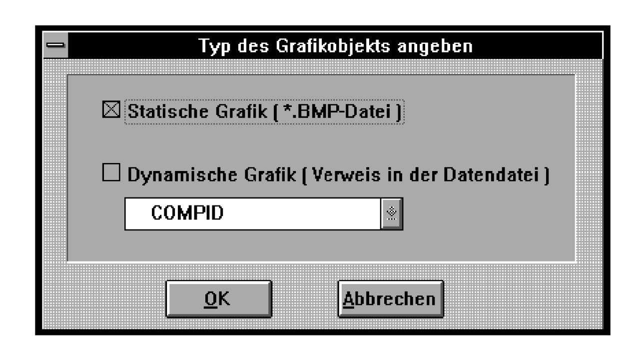


Abb. 5.11: Dialogbox „Typ des Grafikobjekts angeben“

Aktivieren Sie das Kontrollkästchen „Dynamische Grafik“, wird die einzeilige Listbox angezeigt, die alle Feldnamen der aktuellen Datendatei enthält. Nach der Auswahl eines Feldnamens können Sie das Grafikobjekt positionieren. Da das Referenzfeld erst bei der Ausgabe des Reports einen Wert erhält (der sich danach ständig ändern kann), zeigt CodeReporter das dynamische Grafikobjekt auf dem Entwicklungsbildschirm an wie ein normales Feld.

Achtung! Die Größe eines dynamischen Grafikausgabeobjekts muß unbedingt richtig angegeben werden, da CodeReporter die Bitmap so streckt bzw. staucht, daß sie den angegebenen Raum vollständig ausfüllt. Sorgen Sie darüber hinaus dafür, daß alle Bitmaps, auf die das Feld verweist, dieselbe Größe haben. Versäumen Sie das, besteht die Möglichkeit, daß einige Grafikobjekte „korrekt“, während andere verzerrt dargestellt werden.

Grafikobjekte skalieren

CodeReporter bringt Grafikobjekte automatisch auf die Größe des Ausgabeobjekts. Das heißt, größere Bitmap-Bilder werden verkleinert, und kleinere Bitmap-Bilder werden vergrößert.

Diese Anpassung erfolgt auf jeder Achse. Wird ein hohes, schmales Bild in ein quadratisches Grafikobjekt eingepaßt, erhalten Sie ein niedriges, breites Bild. CodeReporter behält die Proportionen des Bildes nicht bei, sondern bringt es auf die Größe des Grafikobjekts.

Größe und Proportionen eines Grafikobjekts werden in derselben Weise verändert wie bei normalen Ausgabeobjekten: entweder in der Dialogbox „Objektdefinition“ oder mit den Größenhenkeln.



Abb. 5.12: Statisches Grafikobjekt

Felder

Feldobjekte spiegeln den Inhalt des zusammengesetzten Datensatzes im Augenblick der Gruppenrücksetzbedingung des Bereiches wider. Müssen ein oder mehrere Felder bearbeitet oder zusammengefaßt werden, um eine „passende“ Ausgabe zu erhalten – zum Beispiel die Zusammenfassung von Feldern mit Vor- und Nachnamen – sollten Sie ein Ausdrucksausgabeobjekt verwenden.

Plazierung

Felder aus der zusammengesetzten Datei werden über die einzeilige Listbox „Feldobjekte“ in den Report eingefügt, die mit der Menüoption **OBJEKT | FELD** oder über die Schaltfläche „Feld“ aufgerufen wird. Man kann Feldobjekte einzeln, aber auch mehrere gleichzeitig plazieren.

Ein einzelnes Feld wird in den Report eingefügt, indem man das Feld in der einzeiligen Listbox „Feldobjekte“ markiert, den Cursor mit der Maus an die gewünschte Stelle bewegt und die linke Maustaste klickt. Hier wird das Feld dann mit voreingestellter Größe und Definition eingefügt.

Werden in der Listbox „Feldobjekte“ mehrere Felder markiert, wird bei der Plazierung im Entwicklungsbildschirm die Dialogbox „Feldgestaltung“ aufgerufen. Wie aus der untenstehenden Abbildung ersichtlich, bietet diese Dialogbox mehrere Option zur Positionierung der Ausgabeobjekte.



Abb. 5.13: Dialogbox „Feldgestaltung“

Vom Einfügepunkt aus gesehen, positioniert CodeReporter die Ausgabeobjekte entweder in einer Reihe nach rechts oder nach unten, je nach Einstellung der Optionsschaltflächen „Einfügerichtung“. Hierbei wird ebenfalls die Einstellung des Kontrollkästchens „Umbruch“ berücksichtigt.

Beim Einfügen der Ausgabeobjekte überprüft CodeReporter, ob ein Objekt über den Rand des Reportsbereichs hinausgeht, in dem die Plazierung stattfindet. In einem solchen Fall wird die Größe dieses Objekts gewöhnlich so verändert, daß es in den Bereich paßt und der Einfügemodus verlassen.

Haben Sie jedoch das Kontrollkästchen „Umbruch“ aktiviert, stellt CodeReporter den Einfügepunkt auf die Ausgangskoordinaten zurück, bewegt sich (je nach Einfügerichtung) nach unten oder nach rechts und setzt die Einfügung der Felder fort. Ist der Bereich weiterhin zu klein (falls CodeReporter die rechte untere Ecke erreicht), wird er möglicherweise in der Höhe erweitert, damit alle Ausgabeobjekte hineinpassen.

CodeReporter erweitert den Bereich in der Höhe, wenn die Schaltfläche „Senkrecht“ markiert, das Kontrollkästchen „Umbruch“ aber nicht aktiviert ist bzw. wenn die Schaltfläche „Waagerecht“ markiert und das Kontrollkästchen „Umbruch“ aktiviert ist. Per Voreinstellung ist „Umbruch“ eingestellt.

Ist das Kontrollkästchen „Bezeichnungen“ ausgewählt, fügt CodeReporter in einer Spalte links von den Feldobjekten Textobjekte mit den jeweiligen Feldnamen ein.

Der Abstand zwischen den eingefügten Feldausgabeobjekten (und Bezeichnungen) wird über die Editierfelder „Abstand senkrecht“ und „Abstand waagerecht“ angegeben. Per Voreinstellung beträgt dieser Abstand 0,25 cm (0,1"); er kann aber auf jeden beliebigen Wert eingestellt werden. Mit Hilfe dieser nützlichen Einrichtung lassen sich Ausgabeobjekte rasch in Reports positionieren, die möglicherweise nicht unter Windows ausgegeben werden und bei denen es auf genaue Positionierung ankommt.

Memofelder

Memofelder – ihr Inhalt wird in einer separaten Memodatei gespeichert – verhalten sich genauso wie andere Felder auch. Folgendes ist dabei zu bedenken:

- Ist das Memofeld leer, wird das Objekt mit einem Leerwert angezeigt.
- Ist das Memofeld kleiner als das Ausgabeobjekt, geht der überschüssige Raum nutzlos verloren.
- Ist das Memofeld größer als das Ausgabeobjekt, werden die überschüssigen Daten ignoriert (siehe „Zeilenumbruch“ weiter oben).

Ausdrücke

dBASE-Ausdrücke, die lediglich ein- oder zweimal im Report erforderlich sind, lassen sich mit einem Ausdrucksausgabeobjekt ausgeben. Oder andersherum, ein Ausdrucksobjekt dient der Ausgabe eines ausgewerteten Ausdrucks im Report.

Enthält eine Datendatei zum Beispiel getrennte Felder für Vor- und Nachnamen, die im Report als „Porombka, Jonas“ erscheinen sollen, könnte man ein Ausdrucksausgabeobjekt mit folgendem Inhalt verwenden:

```
TRIM(DBF->NACH_NAME)+', '+DBF->VOR_NAME
```

Erzeugen

CodeReporter wird durch Auswahl der Menüoption **OBJEKT | AUSDRUCK** oder durch Klicken auf die Schaltfläche „Ausdruck“ in den Einfügemodus für Ausdrucksausgabeobjekte versetzt.

An der Form des Mauscursors können Sie erkennen, daß Sie sich im Einfügemodus befinden. Bewegen Sie den Cursor in einen Reportbereich, und klicken Sie die linke Maustaste, um das Ausdrucksausgabeobjekt zu positionieren.

CodeReporter bittet um Eingabe eines Ausgangsausdrucks für das Objekt in die Dialogbox „Ausdruck eingeben“. (Einzelheiten hierzu entnehmen Sie bitte dem Kapitel „Ausdrücke“). Nachdem Sie einen Ausdruck eingegeben haben, klicken Sie auf „OK“, um die Erzeugung des Objekts abzuschließen.

Berechnungen

Eine Berechnung führt eine zahlen- oder zeichenabhängige Rechenoperation aus, die im Report benutzt wird. Diese Berechnungen erfolgen in Form von dBASE-Ausdrücken. In vielen Fällen fügt eine Berechnung einfach mehrere Datenfelder zusammen, doch kann sie sich auch weitaus komplexer gestalten und mit zusammengesetzten Datenfeldern und/oder dBASE-Funktionen arbeiten. Man kann die Berechnung als eine Art „Kurzform“ für die in ihr enthaltene Rechenoperation betrachten.

Eine Berechnung läßt sich in keinem, einem und vielen Berechnungsausgabeobjekten verwenden. Darüber hinaus kann sie nach ihrer Erstellung in jedem Ausdruck innerhalb des gesamten Reports eingesetzt werden – in Sortier-, Abfrageausdrücken, Ausdrucksausgabeobjekten, Relationsausdrücken usw.

So könnte eine Berechnung anhand der folgenden Formel den Gesamtlohn für einen Angestellten ermitteln:

```
BEZ->REG_STD*ANG->RATE + BEZ->OT_STD*ANG->RATE*1.5
```

Vor- und Nachnamen eines Angestellten könnte man mit der folgenden Formel formatieren:

```
TRIM(ANG->V_NAME)+' '+ANG->N_NAME
```

Handelt es sich bei dem Report um eine einfache Lohnliste, die die Namen und den Bruttolohn der Angestellten aufführt, muß möglicherweise für die Formel keine Berechnung definiert werden – man könnte genauso gut den Ausdruck in das Ausdrucksausgabeobjekt schreiben. Müßte der Report allerdings auch noch die Gesamtsumme aller Gehälter im Berechnungszeitraum oder andere Rechenoperationen in bezug auf die Angestellten enthalten, empfehlen wir den Einsatz einer Berechnung.

Die Verwendung einer Berechnung, anstatt die Rechenoperation mehrere Male in verschiedene Ausdrucksobjekte zu schreiben, verkürzt die Entwicklungszeit; darüber hinaus läßt sich der Report leichter verändern und auf dem neuesten Stand halten.

Hinweis: Wird ein Ausdruck innerhalb eines Reports mehr als einmal verwendet, ist die Erstellung einer Berechnung empfehlenswert.

Eine Berechnung selbst kann wiederum weitere Berechnungen enthalten. Das heißt, Sie können eine Berechnung mit der Definition einer anderen Berechnung verschachteln.

Das Nettogehalt zum Beispiel errechnet sich aus dem Bruttogehalt minus Abzügen. Also kann eine Berechnung des Nettogehalts folgendermaßen aussehen:

```
GES_ANG_BEZ() - ABZUG()
```

Dabei handelt es sich bei **`GES_ANG_BEZ()`** und **`ABZUG()`** um bereits vorhandene Berechnungen.

Berechnungen definieren

Eine Berechnung wird über die Dialogbox „Berechnungsobjekt“ und Klicken auf die Schaltfläche „Neue Ber.“ erstellt. Die Dialogbox ähnelt der in Abb. 7.1 dargestellten.

Die zwei Elemente einer Berechnung sind der Name und der Berechnungsausdruck. Beim Namen handelt es sich um eine Zeichenkette, die die Rechenoperation kurz beschreibt. Diese Bezeichnung erscheint in der Listbox „Berechnungen“ in der Dialogbox „Ausdruck eingeben“. Der Berechnungsausdruck kann in das Editierfeld „Berechnungsausdruck“ eingegeben werden.

Hinweis: Der Name einer Berechnung darf keine Leerzeichen enthalten. Im einem solchen Fall faßt CodeReporter lediglich die Zeichen bis zum ersten Leerzeichen als Namen der Berechnung auf.

Obwohl Berechnungen über das Objektmenü erstellt werden, muß nicht für jede Berechnung auch ein Berechnungsausgabeobjekt vorhanden sein. Es ist allgemeine Praxis, Berechnungen nur innerhalb anderer Berechnungen, Ausdrücke und Summen einzusetzen.

Berechnungen löschen

Die Berechnungen werden in der Dialogbox „Berechnungsobjekt“ gelöscht. Markieren Sie dazu den Namen in der Listbox, und klicken Sie auf die Schaltfläche „Ber. löschen“. Dadurch werden die Berechnung, alle Berechnungen/Summen, in denen sie verwendet wird, und alle Ausgabeobjekte gelöscht, die eine der gelöschten Berechnungen oder Summen benutzen.

Achtung! Das Löschen einer Berechnung kann schnell eine Vielzahl verknüpfter Elemente aus dem Report entfernen. Also gehen Sie dabei vorsichtig vor.

Berechnungsobjekte

Nachdem Sie eine Berechnung erstellt haben, können Sie ein Berechnungsausgabeobjekt positionieren, indem Sie:

1. den Namen der gewünschten Berechnung in der Dialogbox „Berechnungsobjekt“ markieren,
2. den Mauscursor auf die gewünschte Stelle in einem Reportbereich positionieren und

3. die linke Maustaste einmal klicken.

Nach der Plazierung des Berechnungsobjekts verläßt CodeReporter automatisch den Einfügemodus.

Hinweis: Löschen Sie ein Berechnungsausgabeobjekt, wird damit nicht gleichzeitig auch die Berechnung gelöscht, auf die es sich gründet.

Summen

Mit einer Summe hat der Reportentwickler die Möglichkeit, Zahlendaten aus numerischen Berechnungen und numerischen Datenfeldern zusammenzufassen und innerhalb des Reports auszugeben. Im wesentlichen handelt es sich bei einer Summe um ein Ausgabeobjekt, das seinen Wert von einem zusammengesetzten Datensatz zum nächsten behält – wobei der Wert bei jedem zusammengesetzten Datensatz aktualisiert wird.

Summen basieren auf bereits vorhandenen numerischen Berechnungen und numerischen Datenfeldern. Infolgedessen können Mehrfachsummen, die auf denselben Daten beruhen, auch dieselbe Berechnung verwenden.

Im Gegensatz zu Berechnungen müssen Summen mit einem Objekt verbunden sein. Das liegt daran, daß eine Summe ihren Wert an einer bestimmten Stelle und zu einem bestimmten Zeitpunkt innerhalb des Reports erreicht. Die Verkaufssumme für die einzelnen Verkäufer zum Beispiel ist bei jedem anders. Ein Verweis auf diese Summe im Titelbereich des Reports hätte einen nicht vorhandenen Wert zur Folge, da der Summe am Anfang des Reports noch kein Wert zugewiesen wurde.

Ein Summenausgabeobjekt innerhalb reportweiter Ausdrücke, z. B. Abfrageausdrücke, hat keinen Sinn, da der Wert der Summe sich erst bei der Reportausgabe errechnet. In diesem Fall ruft der Versuch, die zusammengesetzte Datendatei aufgrund des Reportinhalts zu begrenzen, einen logischen Widerspruch hervor, da der Report sich eben auf die zusammengesetzte Datendatei gründet.

Am besten bringt man Summen in dem Gruppenfuß unter, der die Daten der Untermenge zusammenfaßt – es sei denn, es handelt sich um eine Vorschausumme; in diesem Fall ist der Gruppenkopf-Bereich angebracht.

Eine Summe erstellen

Ein Summenausgabeobjekt erstellen Sie über die Dialogbox „Summenberechnungen“, das mit der Menüoption **OBJEKT | SUMME** bzw. der Schaltfläche „Summe“ im Report-Entwicklungsbildschirm aufgerufen wird.

Die Dialogbox „Summenberechnungen“ enthält eine Listbox, die die bereits vorhandenen numerischen Berechnungen und Summen umfaßt. Summenausgabeobjekte werden in derselben Weise wie Berechnungen in den Report eingefügt:

- Markieren Sie die gewünschte Berechnung, Summe oder numerische Datendatei in der Listbox.
- Positionieren Sie den Mauscursor auf die gewünschte Stelle des Reportbereichs und
- klicken Sie einmal die linke Maustaste.

Mit der Positionierung wird die Dialogbox „Summe bilden“ (Abb. 5.14) aufgerufen.



Abb. 5.14: Dialogbox „Summe bilden“

Wie Berechnungen lassen sich Summen innerhalb CodeReporters in jedem dBASE-Ausdruck verwenden; von daher braucht jede Summe einen eindeutigen Namen. Diese Bezeichnung ist auf „SUMME*n*“ voreingestellt, wobei es sich bei

n um eine aufsteigende Zahl handelt. Der Name läßt sich dahingehend verändern, daß er die Funktion der Summe deutlich umschreibt.

Wird die Summe innerhalb anderer Ausdrücke verwendet, bestimmt dieser Name, gefolgt von runden Klammern „()“, welche Summe verarbeitet werden soll.

Hinweis: Ist der Wert einer Summe in einem dBASE-Ausdruck erforderlich, muß aber im Report nicht erscheinen, läßt sich das Summenausgabeobjekt „verbergen“, indem man seine Größe so weit reduziert, daß sein Inhalt nicht mehr angezeigt werden kann.

Typen

Die bei der Aktualisierung des Summenwertes von einem zusammengesetzten Datensatz zum nächsten ausgeführte Operation richtet sich nach der Verwendung der Summe. Das Ergebnis einer Summe kann eine arithmetische Summe, ein Durchschnitts-, ein Maximal- oder ein Minimalwert sein.

Eine arithmetische Summe addiert die von der ausgewerteten Berechnung, Summe oder dem numerischen Feld zurückgegebene Zahl zu ihrem bereits vorhandenen Wert hinzu.

Hinweis: Soll eine Summe lediglich die Anzahl der Datensätze zwischen zwei Summenrücksetzbedingungen ermitteln, gründen Sie das Summenausgabeobjekt auf eine Berechnung mit der Konstante eins (1).

Eine Durchschnittssumme bildet den Mittelwert der ausgewerteten Berechnung, Summe oder des numerischen Feldes. Dieser Wert wird ermittelt, indem die Summenwerte der Datensätze durch die Anzahl der betroffenen Datensätze dividiert wird.

Bei einer Maximalsumme wird der größte Wert der ausgewerteten Berechnung, Summe oder des numerischen Feldes ermittelt.

Bei einer Minimalsumme wird der kleinste Wert des ausgewerteten Berechnung, Summe oder des numerischen Feldes ermittelt.

Rücksetzausdruck

Mit dem Summenrücksetzausdruck werden Summen gebildet, die Datenuntermengen innerhalb der zusammengesetzten Datendatei zusammenfassen. Die Summe zum Beispiel, die den Gesamtverkauf eines bestimmten Verkäufers beschreibt, summiert die Verkaufsuntermenge eben dieses Verkäufers.

Im allgemeinen stimmt der Summenrücksetzausdruck mit dem Rücksetzausdruck der Gruppe überein, mit der die Summe logisch verbunden ist. Ein Verkaufsreport kann nach Monaten sortiert und gegliedert werden. Für die Verkäufe eines Monats würde als Summenrücksetzausdruck **`DBF->MONAT`** verwendet, der mit dem Monatsgruppenausdruck identisch ist. Siehe Abb. 5.15.

Der Summenrücksetzausdruck muß für die Gruppe, in der er steht, nicht mit dem Gruppenrücksetzausdruck übereinstimmen und kann eine völlig anders geartete Zurücksetzung bewirken.

Der Rücksetzausdruck bestimmt den Zeitpunkt, wann die Summe auf ihren Ausgangswert zurückgesetzt wird. Er wird für jeden zusammengesetzten Datensatz der zusammengesetzten Datendatei ausgewertet; ändert sich nun der Wert des ausgewerteten Rücksetzausdrucks, wird die Summe zurückgesetzt, damit eine neue Summierung erfolgen kann.

Der Wert, auf den die Summe zurückgesetzt wird, richtet sich nach dem Typ der Summe. Arithmetische und Durchschnittssummen werden auf null zurückgesetzt, während Maximal- und Minimalsummen jeweils auf die größt- bzw. kleinstmögliche Zahl zurückgesetzt werden.

Eine reportweite, oder Gesamtsumme wird unter Verwendung eines leeren Summenrücksetzausdrucks ermittelt. Das heißt, die Summe wird lediglich am Anfang des Reports zurückgesetzt, so daß das Summenobjekt bei seiner Ausgabe im Schlußbereich (bzw. Titelbereich, wenn es sich um eine Vorschausumme handelt) den gesamten Report aufsummiert.

Ein Summenausgabeobjekt, das im selben Bereich wie die zu summierenden Daten plaziert wird, bezeichnen wir als Zwischensumme. Da die Summe laufend bei jeder Gruppenzurücksetzung (und Ausgabe der Ausgabeobjekte innerhalb des Gruppenbereichs) aktualisiert wird, enthält das Summenausgabeobjekt einen neuen Wert.

Eine Summe löschen

Löscht man ein Summenausgabeobjekt, wird ebenfalls die Summe gelöscht, auf der es basiert. Gleichzeitig werden alle anderen Summen, Berechnungen, Ausdrücke und Objekte gelöscht, in denen die Summe enthalten ist. Diese Tatsache kann weitere Löschungen nach sich ziehen und aufgrund ihrer kaskadenartigen Wirkung den Report zerstören. Gehen Sie deshalb bei der Löschung von Summenobjekten mit äußerster Vorsicht vor.

Vorschausummen

Vorschausummen werden zur Ausgabe von zusammengefaßten Untermengen verwendet, bevor die Einzelzeilen tatsächlich ausgegeben werden. Dieses wichtige Konzept, in Verbindung mit einem Anwendungsbeispiel, haben wir im Abschnitt „Vorschau“ weiter oben bereits behandelt.

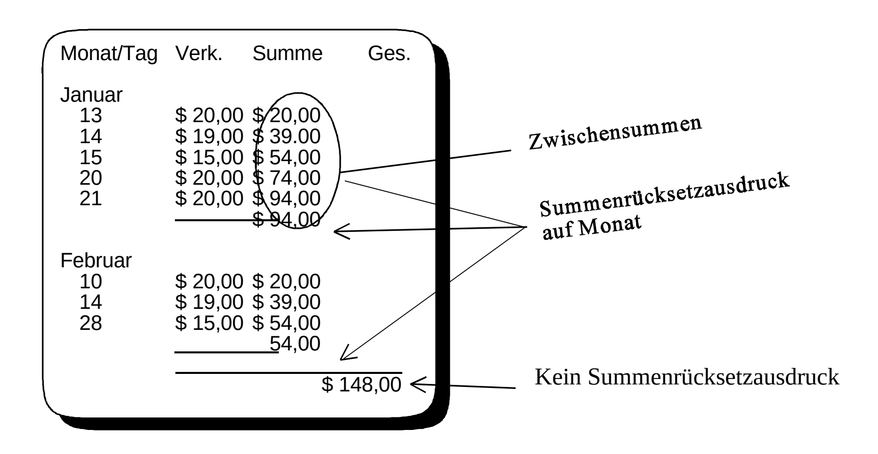


Abb. 5.15: Beispielreport für eine Summe

Bedingte Summen

Man kann Summen mit einem dBASE-Ausdruck verbinden, der bestimmt, unter welchen Bedingungen die Summe ermittelt wird. Durch die Formulierung einer Summierungsbedingung kann das Summenausgabeobjekt seinen Wert aufgrund bestimmter zusammengesetzter Datensätze in der zusammengesetzten Datendatei

aktualisieren, während es andere ignoriert. So soll möglicherweise die Summe lediglich gebildet werden, wenn ein Feld einen bestimmten Wert enthält oder ein Masterdatensatz mit mehreren gefilterten Datensätzen verbunden ist. Durch den Einsatz einer bedingten Summe ist das möglich.

Eine bedingte Summe läßt sich über den Aufruf des Objektmenüs eines Summenausgabeobjekts und die Auswahl der Option **BEDINGTE SUMME** definieren. In die Dialogbox „Bedingte Summe", die dann erscheint (Abb. 5.16), können Sie die Bedingung für die Summe eingeben.

Das Eingabefenster „Summenbedingung" nimmt den dBASE-Ausdruck auf, der den Zeitpunkt der Summenbildung bestimmt. Dieser Ausdruck kann, bis auf `Memo`, jeder beliebige dBASE-Typ sein.



Abb. 5.16: Dialogbox „Bedingte Summe"

Die Schaltflächen „Logische Bedingung" und „Wertveränderung" bestimmen, wie die Summenbedingung bei der selektiven Aktualisierung des Summenwertes eingesetzt wird.

Ist die Schaltfläche „Logische Bedingung" aktiviert und die Summenbedingung (die auf einen logischen Wert hinauslaufen muß) ist logisch wahr, wird die Summe mit dem Inhalt des aktuellen zusammengesetzten Datensatzes aktualisiert. Ist

das Ergebnis des Ausdrucks logisch falsch, wird der aktuelle zusammengesetzte Datensatz ignoriert.

Bei Auswahl der Schaltfläche „Wertveränderung“ wird die Summe lediglich bei einer Änderung der ausgewerteten Summenbedingung aktualisiert. Bei einer Filterrelation zum Beispiel wird der Inhalt der Masterdatei in der zusammengesetzten Datendatei für jeden Datensatz der verknüpften Slavedatei wiederholt. Ein Summenausgabeobjekt, das von einem Feld in der Masterdatei abhängt, würde „nicht richtig“ aufsummiert, da das Feld der Masterdatei mehrmals wiederholt wird.

Die Aktivierung der Schaltfläche „Wertveränderung“ und die Verwendung der Summenbedingung, die dem Wechsel des Datensatzes der Masterdatei entspricht (bzw. dem Masterausdruck in der Filterrelation), sorgen dafür, daß die Summe „richtig“ gebildet wird.

6 Spaltenreport-Utility

Mit der Dialogbox „Spaltenreport-Utility“ steht Ihnen in CodeReporter eine Option zur automatischen Entwicklung von Reports zur Verfügung. Die in Abb. 6.1 gezeigte Dialogbox gestattet Ihnen unter anderem

- das Einfügen von Feldern in das Relationsset,
- das Erzeugen von Gruppenkopf- und -fußbereichen und
- das automatische Bilden von Summen und Zwischensummen bei numerischen Feldern.

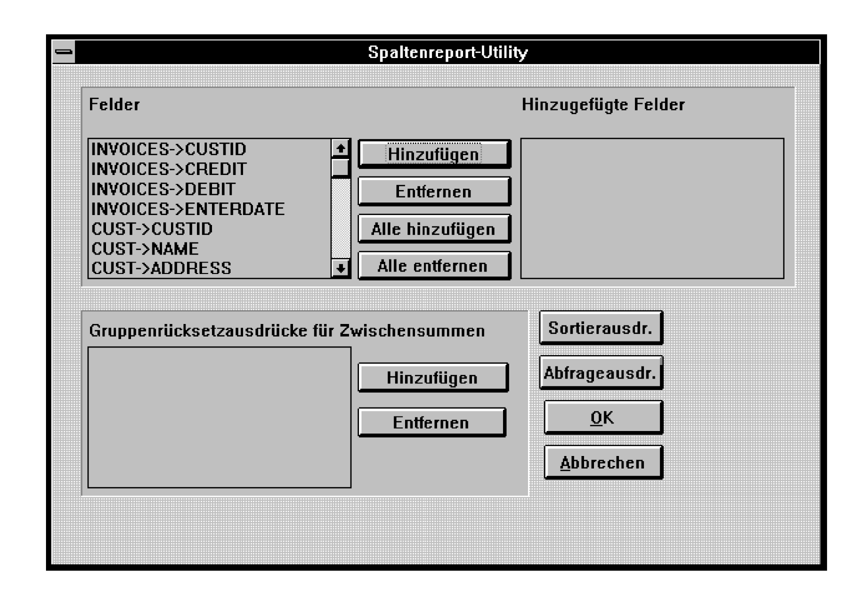


Abb. 6.1: Dialogbox „Spaltenreport-Utility“

Report-Utility aufrufen

Die Dialogbox „Spaltenreport-Utility“ wird mittels der Menüoption **REPORT | SPALTENREPORT-UTILITY** aufgerufen. Enthält der aktuelle Report bereits Ausgabeobjekte, werden diese beim Aufrufen der Dialogbox zerstört. Damit das nicht

geschieht, fordert CodeReporter den Benutzer vorher auf, die Datei zu speichern oder den Spaltenreport abzubrechen.

Einen Report erstellen

Die Dialogbox „Spaltenreport-Utility“ wird dazu verwendet, markierte Felder zum Report hinzuzufügen, Zwischensummen von numerischen Feldern zu bilden und die zusammengesetzte Datendatei abzufragen und zu sortieren.

Nachdem Sie den Report innerhalb der Dialogbox definiert haben, klicken Sie auf „OK“, um den Report zu erzeugen. Klicken Sie auf „Abbrechen“, können Sie die Dialogbox verlassen, ohne daß dabei ein Report erzeugt wird.

Felder hinzufügen

Alle Felder der zusammengesetzten Datendatei werden in der Listbox „Felder“ aufgeführt. Hier können mehrere Felder markiert und nach Klicken auf die Schaltfläche „Hinzufügen“ in den Report übernommen werden. Klicken Sie auf „Alle hinzufügen“, werden automatisch alle Felder der zusammengesetzten Datendatei zum Report hinzugefügt.

Die Felder werden, von links nach rechts, in der Reihenfolge in den Report eingefügt, in der sie in der Listenbox „Enthaltene Felder“ erscheinen. Nehmen die Felder hier nicht die gewünschte Reihenfolge ein bzw. haben Sie ein Feld aus Versehen hinzugefügt, können Sie das/die Feld(er) mit „Entfernen“ oder „Alle entfernen“ aus dem Report löschen.

Hinweis: Haben Sie mehr Felder in den Report eingefügt als in einer Zeile Platz haben, wird das Feld, das über den Rand hinausragt, abgeschnitten, und die weiteren Felder werden in eine zweite Zeile gesetzt.

Zwischensummen

Der über die Dialogbox „Spaltenreport-Utility“ erstellte Report bildet reportweit automatisch Summen für alle numerischen Felder und setzt diese Summen in den Schlußbereich. Sollen Zwischensummen für Untermengen der zusammengesetzten Datendatei gebildet werden, muß die Untermenge definiert werden, mit der sie verbunden sind.

Das geschieht mittels der Schaltfläche „Hinzufügen“ unter „Gruppenrücksetzausdrücke für Zwischensummen“ im unteren Bereich der Dialogbox. Die Schaltfläche ruft die Dialogbox „Gruppenausdruck eingeben“ auf, in die ein Gruppenausdruck für die Untermenge eingegeben werden kann, für die die Zwischensumme gebildet werden soll.

Automatisch erzeugt das Spaltenreport-Utility eine Gruppe (namens „Gruppe *n*“, wobei *n* der Gruppenzähler ist) für den Ausdruck und setzt die Zwischensummen in den Fußbereich der Gruppe.

Hinweis: Die Zwischensummengruppen werden in der Listbox „Subtotal on“ an die erste Stelle gesetzt. Die Gruppen werden also von innen nach außen hinzugefügt.

Haben Sie eine Gruppe an der falschen Position hinzugefügt, können Sie diesen Fehler mit Hilfe der Schaltflächen „Entfernen“ und „Hinzufügen“ bereinigen. Darüber hinaus läßt sich die Position der Gruppe nach der Erzeugung des Reports verändern.

Sortieren und abfragen

In der Dialogbox „Spaltenreport-Utility“ können Sie über die Schaltflächen „Sortierausdr.“ und „Abfrageausdr.“ den Sortier- bzw. Abfrageausdruck für den Report angeben. Beide Schaltflächen rufen zur Eingabe die Dialogbox „Ausdruck eingeben“ auf.

Einzelheiten zum Sortieren und Abfragen der zusammengesetzten Datendatei entnehmen Sie bitte dem Kapitel „Relationale Reports“.

Hinweis: Die Dialogbox „Ausdruck eingeben“ enthält bereits vorhandene Sortier- bzw. Abfrageausdrücke. Sind diese Ausdrücke nicht mehr gültig, können sie einfach aus der Dialogbox gelöscht werden.

Beispiel

Bei dem folgenden Beispiel für einen einfachen Spaltenreport wird für die Firma „ATHA“ ein Verkaufsreport erstellt, der für jedes Geschäft Zwischensummen enthält.

SALES.DBF lokalisieren

Die Datendatei, auf die sich der Verkaufsreport stützt, heißt SALES.DBF und befindet sich im Verzeichnis .\EXAMPLES. Legen Sie über die Menüoption **DATEI | NEU** einen neuen Report an, und wählen Sie in der Dialogbox „Datendatei auswählen“ SALES.DBF aus.

Der Einfachheit halber wird dieser Report nicht mit den Datendateien COMPANY.DBF und STORES.DBF verknüpft. Infolgedessen werden die Daten der für den Verkauf verantwortlichen Firma und des Geschäfts ausgegeben wie in SALES.DBF gespeichert – als Code.

Dialogbox „Report-Utility“

Rufen Sie zunächst die Dialogbox „Spaltenreport-Utility“ über die Menüoption **REPORT | SPALTENREPORT-UTILITY** auf.

Fast alle Felder der Datendatei SALES.DBF sollen in den Report eingehen. Da es beim vorliegenden Report in erster Linie um die finanzielle Seite der Läden und nicht die Verkaufsprodukte geht, kann das Feld PRODCODE unberücksichtigt bleiben.

Die Felder von SALES.DBF (COMPID, STOREID, AMOUNT und PRODCODE) werden in der Listbox „Felder“ aufgeführt. Klicken Sie auf die Schaltfläche „Alle hinzufügen“, um alle in der Listbox „Enthaltene Felder“ zu übernehmen. Markieren Sie das Feld PRODCODE, das nicht in den Report eingehen soll, und klicken Sie auf „Entfernen“; das Feld wird in der Listbox „Enthaltene Felder“ gelöscht und erscheint wieder in der Listbox „Felder“.

Laut Reportzweckerklärung sollen für jedes Geschäft Zwischensummen ausgegeben werden. Über das Feld STOREID wird, wie in Abb. 3.1 über das Feld ORT, eine logische Untermenge der zusammengesetzten Datendatei gebildet. Der Ausdruck, der diese Untermenge beschreibt, lautet:

```
SALES->STOREID
```

(Normalerweise würde der Ausdruck auch die Hauptuntermenge von COMPID mit einschließen; aber da die Firmen mittels einer Abfrage auf „ATHA“ beschränkt werden sollen, reicht der obige Ausdruck völlig aus.)

Rufen Sie über die Schaltfläche „Hinzufügen“ im Bereich „Gruppenrücksetzausdrücke für Zwischensummen“ die Dialogbox „Gruppenausdruck eingeben“ auf, und geben Sie diesen Ausdruck ein.

Der Ausdruck sollte in der Listbox „Gruppenrücksetzausdrücke für Zwischensummen“ angezeigt werden.

Höchstwahrscheinlich wurden die Datensätze in SALES.DBF für die einzelnen Geschäfte nicht sortiert eingegeben, so daß die Datendatei sortiert werden muß. Das geschieht über das Anklicken der Schaltfläche „Sortierausdr.“ und Eingabe desselben:

```
SALES->STOREID
```

Der Sortierausdruck kann das Feld COMPID, die Hauptuntermenge der Datendatei SALES.DBF, gefahrlos vernachlässigen, da lediglich die Verkäufe der Firma „ATHA“ in den Report eingehen sollen.

Die Begrenzung der zusammengesetzten Datendatei erfolgt über den Abfrageausdruck `SALES->COMPID="ATHA"`. Klicken Sie auf „Abfrageausdr.“, und geben Sie den Ausdruck in die Dialogbox „Abfrageausdruck eingeben“ ein.

Den Report einsehen

Nach der Eingabe des Sortier- bzw. Abfrageausdrucks wird der Spaltenreport erzeugt, indem Sie auf „OK“ klicken.

Lassen Sie sich den Report mittels der Menüoption **DATEI | DRUCKBILD EINSEHEN** anzeigen. Die Felder und die Summen werden im Ausgabefenster korrekt ausgegeben.

Ausfeilen

Der vorliegende Schnellreport ist weit davon entfernt, gedruckt zu werden. Um ihn „auszufeilen“, stehen Ihnen verschiedene Möglichkeiten zur Verfügung, als da sind: unterschiedliche Styles und Schriften, Erläuterungen zu den Summen, Reporttitel usw. Diese Änderungen können am Schnellreport genauso vorgenommen werden wie bei einem Report, der standardmäßig erstellt worden ist.

7 Ausdrücke

Ein Großteil der Kommunikation zwischen CodeReporter und dem Reportentwickler erfolgt über den Einsatz von dBASE-Ausdrücken. Dabei handelt es sich, vom Begriff her, um eine Makrosprache, mit der Vorgänge beschrieben bzw. Daten identifiziert werden.

dBASE-Ausdrücke verhalten sich in vielerlei Hinsicht wie mathematische Gleichungen. Werte können zusammengezogen, Gleichungen können auf Richtigkeit (wahr oder falsch) überprüft werden usw. Wie in der Mathematik gelten für dBASE-Ausdrücke bestimmte Regeln. Im Gegensatz zur Mathematik, bei der lediglich mit Zahlenwerten operiert wird, können dBASE-Ausdrücke auch andere Werte annehmen, z. B. alphanumerisch und Datum.

Leser, denen dBASE bzw. dBASE-Ausdrücke bereits ein Begriff sind, können die nächsten Abschnitte überspringen und bei „Kurz-Ausdruck eingeben“ weiterlesen.

Allgemeine Hinweise zu dBASE-Ausdrücken

Alle dBASE-Ausdrücke geben einen Wert eines bestimmten Typs zurück. Dieser Typ kann „Numerisch“, „Zeichen“, „Datum“ oder „Logisch“ sein.

Eine geläufige Form eines dBASE-Ausdrucks ist der Name eines Feldes. In diesem Fall stimmt der Typ des dBASE-Ausdrucks mit dem Typ des Feldes überein.

Feldnamen, Konstanten und Funktionen können als Teile von dBASE-Ausdrükken benutzt werden. Diese Teile lassen sich mit anderen Funktionen oder mit Operatoren kombinieren.

Beispiel für einen dBASE-Ausdruck:

```
UPPER(DBF->FELD_NAME)
```

In der Mathematik wird „1+2“ als eine Aussage betrachtet, die denselben Wert aufweist wie „3“. „4x3÷6“ entspricht „2“. In der dBASE-Terminologie geben diese mathematischen Aussagen einen Wert zurück. Die Zahlen werden auf mathematisch richtige Weise zusammengefaßt, und das Ergebnis wird gebildet.

Würde ein Schüler einen Richtig-oder-falsch-Test schreiben und er/sie müßte die Gleichung 1+2=5 auswerten, lautete die richtige Antwort „falsch“. Bei 1+2=3 lautete die Antwort dagegen „richtig“. Die Werte auf jeder Seite des Gleichheitszeichens werden getrennt ausgewertet und ihre Rückgabewerte verglichen. Im ersten Fall ergeben 1+2 3, und 5 hat den Wert 5. Die Aussage, daß 3 mit 5 übereinstimmt, ist nicht richtig, also falsch. Bei 1+2=3 jedoch ergeben beide Seiten 3. dBASE-Ausdrücke, bei denen beide Seiten der Gleichung vorhanden sind, bezeichnen wir als logische Ausdrücke. Diese Ausdrücke bedienen sich sogenannter Vergleichsoperatoren, die wir weiter unten anführen.

Kennzeichner von Feldnamen

Da in die meisten Reports eine Vielzahl von Datendateien eingeht, ist es erforderlich, einen Feldnamen in einem dBASE-Ausdruck durch die Angabe der Datendatei zu kennzeichnen.

Der erste Teil eines Feldnamens, der Kennzeichner, bezeichnet ein Datendatei-Alias. Dabei handelt es sich normalerweise einfach um den Namen der Datendatei, dem nach dem „->“ der eigentliche Feldname folgt.

dBASE-Ausdruckskonstanten

dBASE-Ausdrücke können aus numerischen, logischen oder Zeichenkonstanten bestehen. Solche Ausdrücke sind allerdings meistens nicht sehr nützlich. Konstanten werden gewöhnlich in komplizierteren dBASE-Ausdrücken benutzt.

Eine numerische Konstante ist eine Zahl. Zum Beispiel sind „5“, „7.3“ und „18“ dBASE-Ausdrücke mit numerischen Konstanten.

Zeichenkonstanten sind in Anführungszeichen eingeschlossene Buchstaben. 'Das sind Daten', 'Hans Schmidt' und '"Hans Schmidt"' sind Beispiele für dBASE-Ausdrücke mit Zeichenkonstanten. Wenn Sie eine Zeichenkonstante definieren möchten, die einfache oder doppelte Anführungszeichen enthält, benutzen Sie den jeweils anderen Typ von Anführungszeichen, um die Zeichenkonstante zu markieren. "Peter's" und '"OK"' z. B. sind gültige Zeichenkonstanten.

Hinweis: Wenn nicht anders angegeben, sind in diesem Handbuch alle dBASE-Zeichenkonstanten in einfache Anführungszeichen eingeschlossen.

Die Konstanten .TRUE. und .FALSE. sind die einzigen gültigen logischen Konstanten. Die Konstanten .T. und .F. sind gültige Abkürzungen dafür.

Eine Datumskonstante kann unter Verwendung der dBASE-Funktion STOD() mit einer Zeichenkonstante als Parameter ermittelt werden.

Operatoren der dBASE-Ausdrücke

Mit Operatoren wie „+“, „*“ oder „<“ werden Konstanten und Felder bearbeitet. „3+8“ ist ein Beispiel für einen dBASE-Ausdruck, bei dem der Additionsoperator auf zwei numerische Konstanten wirkt, um den numerischen Wert „11“ zurückzugeben.

Die Werte, auf die ein Operator wirkt, müssen einen mit dem Operator übereinstimmenden Typ aufweisen. Zum Beispiel wirkt der Divisionsoperator „/“ auf zwei numerische Werte.

Rangfolge

Die Rangfolge der Operatoren bestimmt die Reihenfolge, in der die Operatoren ausgewertet werden. Die Rangfolge der Operatoren wird in den folgenden Tabellen angegeben. Je höher der Rang, desto früher wird die Operation ausgeführt. Beispielsweise nimmt „dividieren“ Rang 6, „addieren“ Rang 5 ein; das bedeutet, daß „dividieren“ vor „addieren“ ausgewertet wird. Infolgedessen ist das Ergebnis von „1+4/2“ gleich „3“.

Die Reihenfolge der Auswertung kann durch den Einsatz von Klammern verdeutlicht werden. Zum Beispiel „1+2 * 3“ ergibt „7“, während „(1+2) * 3“ „9“ ergibt.

Name des Operators	Symbol	Vorrang
Addition	+	5
Subtraktion	-	5
Multiplikation	*	6
Division	/	6
Potenzrechnung	** oder ^	7

Es gibt zwei Zeichenoperatoren, „Verketten I“ und „Verketten II“ genannt, die zwei Zeichenwerte zu einem verbinden. Sie unterscheiden sich dadurch von den Operatoren Addition und Subtraktion, daß sie bei anderen Werttypen, nämlich Zeichenketten, eingesetzt werden.

Name des Operators	**Symbol**	**Vorrang**
Verketten I	+	5
Verketten II	-	5

Beispiele   "'Hans '+'Schmidt'"   wird zu   "'Hans Schmidt'"
              "'ABC'+'DEF'"   wird zu   "'ABCDEF'"

Verketten II arbeitet ähnlich, hängt aber alle endständigen Leerzeichen der ersten Zeichenkette an das Ende des Gesamtausdrucks an.

Beispiele   "'Hans'-'Schmidt '"   wird zu   "'HansSchmidt '"
              "'ABC'-'DEF'"   wird zu   "'ABCDEF'"
              "'A '"-'D '"   wird zu   "'AD '"

Vergleichsoperatoren geben ein logisches Ergebnis (entweder wahr oder falsch) zurück. Alle Operatoren mit der Ausnahme von „Beinhalten“ arbeiten mit numerischen, Zeichen- oder Datumswerten. „Beinhalten“ arbeitet mit zwei Zeichenketten und gibt wahr zurück, wenn die erste in der zweiten enthalten ist.

Name des Operators	**Symbol**	**Vorrang**
Gleich	=	4
Ungleich	<> oder #	4
Kleiner als	<	4
Größer als	>	4
Kleiner oder gleich	< =	4
Größer oder gleich	> =	4
Beinhalten	$	4

Beispiele	"'CD'$'ABCD'"	gibt wahr zurück
	"8<7"	gibt falsch zurück
	"5 = 4 + 1"	gibt wahr zurück

Logische Operatoren geben ein logisches Ergebnis zurück und arbeiten mit zwei logischen Werten.

Name des Operators	**Symbol**	**Vorrang**
Nicht	.NOT.	3
Und	.AND.	2
Oder	.OR.	1

Beispiele	".NOT. .T."	gibt „.T.“ zurück
	".T. .AND. .F."	gibt „.F.“ zurück

Kurz-Ausdruck eingeben

An vielen Stellen bittet CodeReporter den Reportentwickler um die Eingabe von dBASE-Ausdrücken. So erfordern zum Beispiel das Sortieren, Abfragen und Verknüpfen von Datendateien immer wieder einen oder mehrere dBASE-Ausdrücke.

Ist ein solcher Ausdruck erforderlich, bietet CodeReporter im allgemeinen zwei Möglichkeiten, ihn zu definieren: entweder durch direkte Eingabe oder über die Dialogbox „Ausdruck eingeben“, deren Titelleiste je nach Kontext variiert.



Abb. 7.1: Beispiel für die Schaltfläche „Kurz-Ausdr.“

Die obige Abbildung ist ein typisches Beispiel für die Anforderung eines dBASE-Ausdrucks. Die Dialogbox „Berechnung“ enthält einen Bereich (das Editierfeld „Berechnungsausdruck“), in den der Ausdruck unmittelbar eingegeben werden kann. Stellen Sie den Cursor direkt in diesen Bereich, und geben Sie den entsprechenden dBASE-Ausdruck ein.

Des weiteren enthält die Abb. 7.1 eine Aufrufmöglichkeit der Dialogbox „Ausdruck eingeben“ – die Schaltfläche „Kurz-Ausdr.“. In dem besagten Dialog brauchen Sie die entsprechenden dBASE-Ausdrücke dann nur noch anzuklicken. Sind Sie damit fertig, wird der in der Dialogbox „Ausdruck eingeben“ vorhandene Ausdruck in das Editierfeld „Berechnung“ übernommen.

Dialogbox „Ausdruck eingeben“

Die Dialogbox „Ausdruck eingeben“ (Abb. 7.2) stellt dem Reportentwickler eine einfache Möglichkeit zum Aufbau eines dBASE-Ausdrucks zur Verfügung. Im Editierfeld „Ausdruck“ wird der Ausdruck nach und nach aufgebaut. Doppelklicken Sie auf ein Element der vorhandenen Listboxen, wird es an der aktuellen Schreibmarkenposition in den Ausdruck übernommen. Die dBASE-Operatoren lassen sich über einen einfachen Klick auf die gewünschte Schaltfläche hinzufügen.

Die Dialogbox „Ausdruck eingeben“ bietet beim Aufbau eines Ausdrucks einen erheblichen Geschwindigkeitsvorteil. Die Felder werden automatisch mit ihren Dateikennzeichnern hinzugefügt, während Funktionen automatisch die erforderlichen Klammern und Kommata erhalten usw. Die Listboxen enthalten alle Fel

der der zusammengesetzten Datendatei, jede unterstützte dBASE-Funktion und alle bereits vorhandenen Berechnungen und Summen.

Darüber hinaus sind per Mausklick für alle Felder bzw. Funktionen entsprechende Informationen abrufbereit. Diese werden unterhalb der Listboxen angezeigt. In Abb. 7.2 enthält dieses Feld Informationen über die Funktion TRIM().

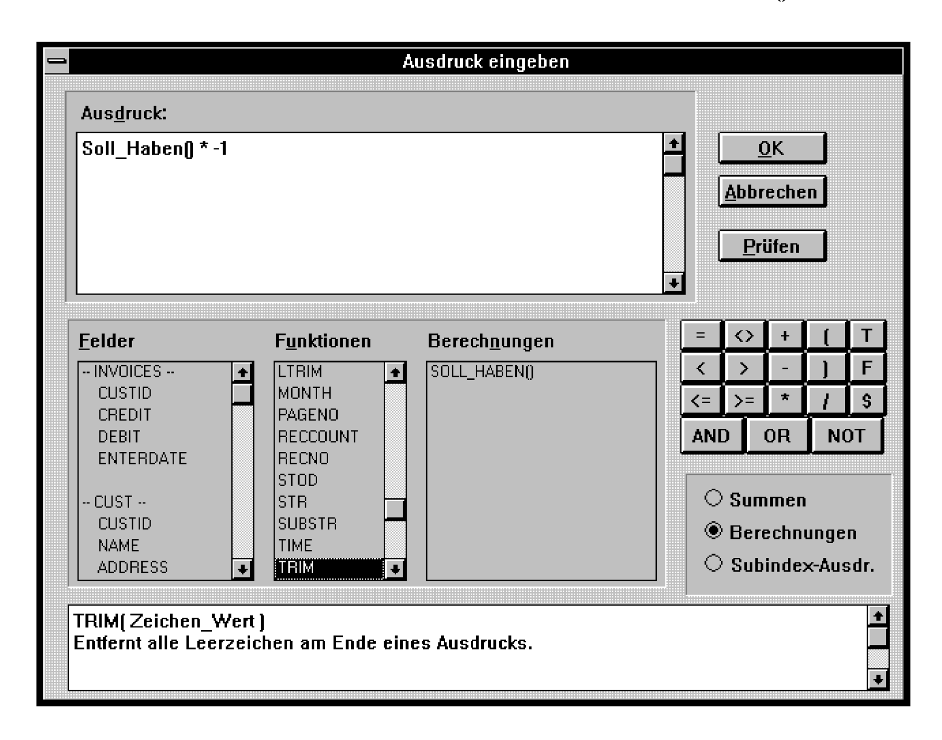


Abb. 7.2: Dialogbox „Ausdruck eingeben“

Verwendung von „Ausdruck eingeben“

Die Arbeit mit der Dialogbox „Ausdruck eingeben“ ist nicht schwer. Die Schaltflächen bieten Ihnen eine vollständige Point-and-klick-Umgebung.

Ins Editierfeld „Ausdruck“ können Sie einen Ausdruck manuell eingeben bzw. einen Ausdruck oder den Teil eines Ausdrucks editieren. Die übrigen Schaltflächen fügen ihre Elemente an der Position der blinkenden Schreibmarke ein, wenn

die Listbox angewählt ist.

Die Listbox „Felder" enthält alle Felder der zusammengesetzten Datendatei. Die Felder der verschiedenen Datendateien werden durch „--*xxxxxxx*--" voneinander getrennt, wobei *xxxxxxx* jeweils der Name einer Datendatei ist.

Wird ein Feld mit einem einfachen Mausklick angewählt (oder die Markierung in die Listbox bewegt), werden im Informationsfenster Hinweise zum Feld angezeigt.

Doppelklicken Sie mit der Maus auf ein Feld, wird es (zusammen mit dem Kennzeichner) in das Editierfeld „Ausdruck" übernommen.

Die Listbox „Funktionen" enthält alle unterstützten dBASE-Funktionen, die in einen Ausdruck übernommen werden können.

Bei der Auswahl einer Funktion zeigt das Informationsfenster eine kurze Beschreibung der Funktion an, u. a. Parameter (falls vorhanden) und den Typ der Rückgaben.

Ein Doppelklick auf eine Funktion fügt diese in das Editierfeld „Ausdruck" ein und stellt die Schreibmarke zwischen die Klammern der Funktion.

Die Listbox „Berechnungen" enthält alle bereits vorhandenen Reportberechnungen. Die in den Ausdruck eingefügte Berechnung wird bei der neuen Auswertung berücksichtigt.

Es ist auch möglich, eine Berechnung in die Ausdrücke anderer Berechnungen oder Summen einzuschließen. Mit dieser „Verschachtelung" können Sie effektive Reports erstellen.

In dieser Listbox werden, je nach aktivierter Schaltfläche rechts, (anstelle der Berechnungen) auch die Summen oder Subindex-Ausdrücke angezeigt.

Achtung! Subindex-Ausdrücken fehlen die für CodeReporter-Ausdrücke erforderlichen Feldnamen-Kennzeichner. Soll ein Subindex-Ausdruck verwendet werden, müssen Sie die Kennzeichner per Hand eingeben.

Mit Hilfe der Schaltfläche „Prüfen" können Sie einen in das Editierfeld „Ausdruck" eingegebenen Ausdruck überprüfen lassen. Dabei wird versucht, den Ausdruck auszuwerten. Schlägt der Versuch aus irgendeinem Grund fehl – z. B. nicht kompatible Typen, falsche Parameteranzahl bei einer Funktion oder unbekannte Symbole –, erfolgt eine Fehlermeldung. Andererseits zeigt CodeReporter auch an, wenn der Ausdruck keine Fehler enthält.

Nach der Vervollständigung und Prüfung eines Ausdrucks klicken Sie auf „OK“, um die Dialogbox „Ausdruck eingeben“ zu schließen. Klicken Sie statt dessen auf „Abbrechen“, verlassen Sie zwar die Dialogbox, aber alle Ihre Eingaben gehen verloren.

8 Styles

Ein professionell gestalteter Report enthält nicht nur die erforderlichen Informationen, sondern präsentiert sie auch in ansprechender Weise. Einen wichtigen Aspekt dabei bilden Schriftart, -größe und Farbe, mit deren Hilfe wichtige Ausgabeobjekte (Spaltentitel, Titel, Summen, negative Zahlen usw.) hervorgehoben werden können.

In CodeReporter läßt sich eine derartige Formatierung über die Verwendung von Styles verwirklichen. Mit einem Style legen Sie Schriftart, -größe, Farbe und Sonderattribute (unterstreichen, fett, kursiv usw.) fest und geben ihm einen Namen. Jedesmal wenn ein Ausgabeobjekt gleichartig strukturiert sein soll, laden Sie den entsprechenden Style; es ist nicht erforderlich, das Erscheinungsbild immer wieder neu aufzubauen.

Im Laufe der Reportentwicklung können Sie neue Styles definieren, um den Report Ihren Anforderungen anzupassen.

Warum Styles?

Aus verschiedenen Gründen setzt CodeReporter Styles sowie Seiten-Layouts ein. In erster Linie soll dadurch die Gesamtanlage des Reports erleichtert werden. Das Erscheinungsbild des Reports (oder das mehrerer Reports) braucht lediglich einmal definiert zu werden. Nach seiner Fertigstellung müssen Sie sich nicht noch einmal Gedanken über die Verwendung bestimmter Schriftarten machen.

Darüber hinaus fördern Styles das gleichmäßige Aussehen des Reports. Anstatt für jedes Ausgabeobjekt immer wieder neu Schriftart, -größe und Farbe festlegen zu müssen (und dabei möglicherweise Fehler zu begehen), muß ein Style für Objekte, die eine ähnliche Aussage vermitteln sollen, lediglich einmal definiert werden. Sollen zum Beispiel alle Summenobjekte in einer bestimmten Farbe erscheinen, legen Sie die Farbe einfach innerhalb eines Styles fest und verbinden diesen mit den Summenobjekten.

Durch den Einsatz von Seiten-Layouts lassen sich gemeinsame Styles in mehreren Reports verwenden. CodeReporter speichert die Style-Daten in einer besonderen Datei ab, die in andere Reports geladen werden kann. Auf diese Weise

können verschiedene Reports dasselbe Erscheinungsbild aufweisen, ohne daß Sie die Styles jeweils neu definieren müssen.

Die Flexibilität von Styles und Seiten-Layouts wird offensichtlich, wenn Sie das Aussehen eines Reports ändern müssen. Soll eine Schriftart oder eine Farbe im gesamten Report verändert werden, brauchen Sie lediglich den Style zu bearbeiten, und alle Ausgabeobjekte innerhalb des Reports werden automatisch aktualisiert.

In Verbindung mit weiteren Produkten von Sequiter und dem CodeReporter-API (Application Programming Interface) lassen sich mit CodeReporter unter Windows erstellte Reports auch unter anderen Betriebssystemen (DOS, Unix, OS/2 usw.) ausgeben. Die Ausgabe unter diesen Betriebssystemen ist gewöhnlich mit einigen Problemen verbunden.

Für diesen Fall stellt CodeReporter Non-Windows-Styles zur Verfügung. Einzelheiten zur Ausgabe von Reports unter anderen Betriebssystemen entnehmen Sie bitte dem Abschnitt „Non-Windows-Styles“ weiter unten.

Styles erstellen

Die Ausgabe neu erstellter Objekte erfolgt unter Verwendung des voreingestellten Styles. Wurde bislang kein Style erstellt, gilt der CodeReporter-Grundstyle „Normal“. Sind bereits Styles vorhanden, wird der zuletzt ausgewählte als Voreinstellung für alle neuen Ausgabeobjekte benutzt.

Styles werden über die Menüoption **STYLE | ERSTELLEN** definiert. Nach der Auswahl bittet CodeReporter Sie um die Eingabe eines Namens für den neuen Style, der in dem Report nur einmal vorkommen darf. Ist ein Style mit diesem Namen bereits vorhanden, kann der neue nicht erstellt werden.

Nachdem Sie einen Namen für den Style eingegeben haben, wird die gemeinsame Dialogbox „Schriftart“ (Abb. 8.1) angezeigt, wie sie die meisten Windows-Anwendungen, z. B. Textverarbeitungsprogramme, enthalten.

Wählen Sie die gewünschte Schriftart, -größe, Farbe usw. aus, und klicken Sie auf „OK“. Wenn Sie jetzt ein Ausgabeobjekt mit diesem neuen Style verbinden, erfolgt die Ausgabe mit den eingestellten Schriftattributen.

Hinweis: Beim Erstellen neuer Styles bildet der aktuelle Style die Grundlage für den neuen. Wenn Sie mehrere ähnliche Styles erstellen wollen, definieren Sie den ersten, wählen ihn aus und benutzen ihn dann als Grundlage für die weiteren Styles.

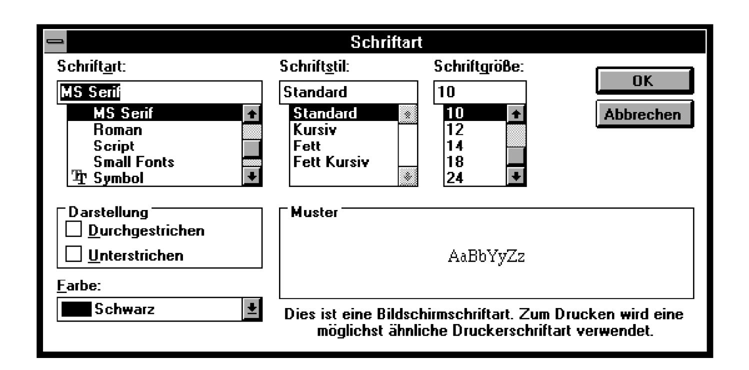


Abb. 8.1: Dialogbox „Schriftart“

Einen Style löschen

Mit Hilfe der Menüoption STYLE | LÖSCHEN lassen sich Styles entfernen. Sind Styles vorhanden, die gelöscht werden können, wird ein Untermenü mit einer Styles-Liste angezeigt. Die Styles, die Sie hierin auswählen, werden gelöscht.

Hinweis: Wenn Sie versuchen, einen Style zu löschen, der zur Zeit mit Ausgabeobjekten verbunden ist, gibt CodeReporter eine Meldung aus.

Ausgabeobjekte, deren Style gelöscht worden ist, werden neu mit dem ersten im Style-Popup-Menü vorhandenen Style verbunden. (Einzelheiten zum Style-Popup-Menü finden Sie weiter unten.)

Hinweis: Für jeden Report muß zumindest ein Style definiert werden. Infolgedessen kann der einzige Style eines Reports auch nicht gelöscht werden.

Einen Style bearbeiten

Über Auswahl der Menüoption **STYLE | BEARBEITEN** können Sie den aktuellen Style verändern. Dazu wird die gemeinsame Dialogbox „Schriftart“ aufgerufen, in der die gewünschten Änderungen vorgenommen werden.

Wird der aktuelle Style verändert, werden alle Ausgabeobjekte aktualisiert, die diesen Style verwenden. Einzelheiten darüber, wie man einen Style auswählt, entnehmen Sie bitte dem folgenden Abschnitt.

Hinweis: Verwenden Sie für einen Style eine größere Schrift, wirkt sich das nicht auf die Größe der entsprechenden Ausgabeobjekte aus. Möglicherweise wird der Text für das Objekt nicht mehr vollständig angezeigt. Passen Sie das Ausgabeobjekt per Hand an die neue Größe an.

Einen Style auswählen

Einen Style können Sie mit Hilfe der Schaltfläche „Style“ auswählen. (Wo sich diese Schaltfläche befindet, entnehmen Sie bitte dem Kapitel „Starten mit Code-Reporter“.) Wenn Sie auf die Schaltfläche klicken, wird Ihnen eine Popup-Liste aller für den Report erstellten Styles angezeigt.

Aus dieser Liste können Sie einen Style per Einfach- bzw. Doppelklick auswählen. Im Falle eines Einfachklicks müssen Sie die Schaltfläche noch einmal anklicken, um das Style-Popup-Menü zu schließen.

Wählen Sie ein Ausgabeobjekt aus, wird damit ebenfalls der mit diesem Objekt verbundene Style ausgewählt.

Für ein Objekt

Sind bei dieser Styleauswahl Ausgabeobjekte markiert, werden Sie mit dem neuen Style verbunden. Auf diese Weise läßt sich der Style für ein Ausgabeobjekt am schnellsten einstellen.

Darüber hinaus läßt sich ein Style über die einzeilige Listbox „Style“ in der Dialogbox „Objektdefinition“ einstellen. Einzelheiten zu dieser Dialogbox entnehmen Sie bitte dem Abschnitt „Objekte bearbeiten“ im Kapitel „Objekte“.

Non-Windows-Styles

Die Reportausgabe unter Windows erfolgt unter Verwendung der verschiedenen Windows-Bildschirm- und Druckertreiber. Dabei sieht ein auf einem Nadel- bzw. Laserdrucker ausgegebener Report in etwa gleich aus. Ob ein Drucker eine bestimmte Schrift unterstützt oder nicht, spielt dabei keine Rolle, da Windows die Schrift emuliert – oder den Text als Grafik ausgibt. Dieses Leistungsmerkmal fehlt den meisten Non-Windows-Anwendungen.

Hinweis: Non-Windows-Styles sind lediglich erforderlich, wenn der Report unter einem anderen Betriebssystem ausgegeben werden soll. Soll er einzig und allein unter Windows verarbeitet werden, brauchen Sie die Non-Windows-Styles nicht zu berücksichtigen.

Bei Non-Windows-Anwendungen können lediglich die Schriften und Attribute gedruckt werden, die der Drucker unterstützt, da die Ausgabe ausschließlich über den Drucker läuft. Um noch mehr Verwirrung zu stiften: verschiedene Drucker aktivieren dieselben Attribute (z. B. Fettdruck) auf unterschiedliche Art und Weise. So sind bei einem vielleicht Paare von Steuerzeichen erforderlich, während ein zweiter etwas völlig anderes benötigt.

Durch die Verwendung von Non-Windows-Styledefinitionen können druckerspezifische Steuerzeichen – die den Drucker den Text in unterschiedlichen Schriften ausgeben lassen – in die Styles eines Seiten-Layouts integriert werden. Der Vorteil dieser Methode besteht darin, daß man für die verschiedenen Drucker eigene Seiten-Layouts erstellen kann, die bei der Ausgabe des Reports unter dem anderen Betriebssystem jeweils geladen werden. Je nach Druckertyp (z. B. Laserdrucker, Nadeldrucker) werden auf diese Weise die korrekten Steuersequenzen in die Styles eingebaut.

Den API-Funktionen von CodeReporter sind die Steuerzeichen als solche oder das geladene Seiten-Layout gleichgültig. Einzelheiten hierzu finden Sie weiter unten.

Nähere Einzelheiten zur Ausgabe von Reports in Non-Windows-Betriebssystemen entnehmen Sie bitte der Funktionsübersicht.

Styles angeben

Wie bereits erwähnt, gibt es druckerspezifische Steuersequenzen, aufgrund derer der Drucker Texte mit unterschiedlichen Attributen ausgibt – z. B. in Fett.

Die meisten Drucker verwenden zur Steuerung der relevanten Funktionen oder Attribute zwei Steuersequenzen: die eine, um die Funktion „einzuschalten“, die andere, um sie wieder „auszuschalten“. Die Aktivierung von Fettschrift könnte ESC "G+" sein, während der Drucker mit ESC "G-" wieder auf seinen Standardmodus zurückgesetzt wird. Die für eine Schriftart oder ein Attribut erforderlichen Steuersequenzen hängen von dem Drucker ab, über den die Reportausgabe unter dem Non-Windows-Betriebssystem erfolgen soll. Sie finden sie üblicherweise in Ihrem Druckerhandbuch.

Die API-Funktionen von CodeReporter schicken die Einleitungssequenz vor dem Text der Ausgabeobjekte auf den Drucker und schließen die Ausgabe mit der Ausleitungssequemz ab.

Die Steuersequenzen für den aktuellen Style können Sie mit Hilfe der Menüoption **STYLE | DEFINITION FREMDFORMAT** eingeben. Die Option ruft die Dialogbox „Informationen über Fremdformat-(Non-Windows)Layout“ auf (Abb. 8.2).

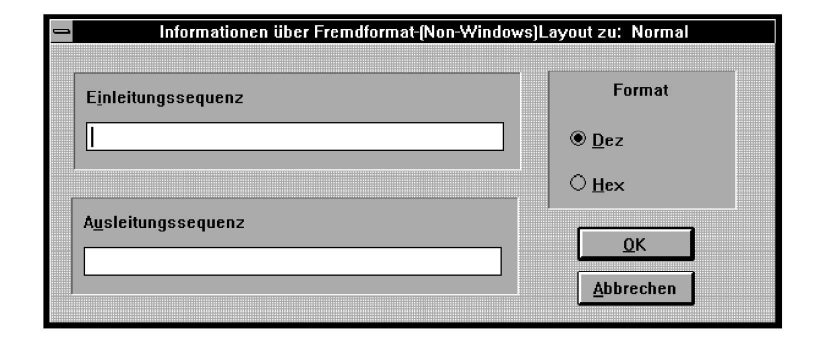


Abb. 8.2: Dialogbox „Informationen über Fremdformat-(Non-Windows)Layout“

Im Editierfeld „Einleitungssequenz“ geben Sie die Steuerzeichen an, die eine bestimmte Druckerschrift aktivieren, während das Editierfeld „Ausleitungssequenz“ die Steuerzeichen aufnimmt, die das Attribut deaktivieren.
Das Format der Steuersequenz richtet sich nach den Schaltflächen unter „Format Steuersequenz“. Ist die Schaltfläche „Dez“ aktiv, wird der in das Editierfeld eingegebene Text als Dezimalwert interpretiert; ist die Schaltfläche „Hex“ aktiv,

muß der eingegebene Text hexadezimalen Werten entsprechen. Zwischen den einzelnen Steuersequenzen muß jeweils ein Leerzeichen stehen. Das Leerzeichen wird nicht an den Drucker geschickt, sondern dient lediglich als Begrenzer der einzelnen Steuersequenzen.

Mehrfachattribute für Non-Windows-Ausgabeobjekte, z. B. fett und kursiv, müssen innerhalb einer Non-Windows-Styledefinition angegeben werden. Setzen Sie dabei jeweils ein Leerzeichen zwischen Ende und Anfang der Steuersequenzen. Ob bei Ihrem Drucker eine bestimmte Reihenfolge bei kombinierten Steuersequenzen erforderlich ist, entnehmen Sie bitte Ihrem Druckerhandbuch.

Wie bereits erwähnt, sind die Werte für Schriftarten und Attribute von Drucker zu Drucker verschieden. In Anhang D, Auszug aus der ASCII-Tabelle, finden Sie die entsprechenden Umrechnungswerte für ASCII-, Hexadezimal- und Dezimalnotation.

Beispiel

Das folgende Beispiel zeigt die erforderlichen Schritte zur Erstellung einer Kursivschrift (sowohl für Windows als auch ein Non-Windows-Betriebssystem) und erklärt, wie man den neuen Style mit einem bereits vorhandenen Ausgabeobjekt verbindet. Dabei gehen wir davon aus, daß es sich bei dem Non-Windows-Drukker um ein Epson-kompatibles Gerät handelt.

Wählen Sie die Menüoption **DATEI | NEU** aus, um einen neuen Report zu beginnen. Da hierbei keine bestimmte Datendatei erforderlich ist, können Sie eine beliebige Datei aus dem Verzeichnis .\EXAMPLES zum Top Master erklären.

Automatisch erstellt CodeReporter die voreingestellte Gruppe „Body“ (mit einem Kopfbereich) mit dem Standard-Style „Normal“.

Versetzen Sie CodeReporter über die Menüoption **OBJEKT | TEXT** in den Einfügemodus für Textobjekte und positionieren zwei Textausgabeobjekte mit folgendem Text in den Gruppenkopf-Bereich:

- Dieser Text ist NICHT kursiv.
- Dieser Text IST kursiv.

Zu diesem Zeitpunkt sind beide Textobjekte noch mit dem voreingestellten Style „Normal“ verbunden, der nicht kursiv ist.

Wählen Sie die Menüoption **STYLE | ERSTELLEN** an, um einen neuen Kursiv-Style zu erzeugen. Wenn Sie zur Eingabe der Stylebezeichnung aufgefordert werden, geben Sie einen sprechenden Namen, z. B. „Kursiv“, ein und klicken auf „OK“.

CodeReporter ruft nun die Dialogbox „Schriftart“ auf (Abb. 8.1), wobei die aktuelle Schrift als Ausgangsschrift für die neue benutzt wird. Um die Schrift kursiv zu gestalten, markieren Sie in der Listbox „Schriftstil“ den Eintrag „Kursiv“ und klicken auf „OK“.

Durch das Erstellen eines neuen Styles werden die Objekte innerhalb des Reports nicht verändert. Deshalb müssen die gewünschten Ausgabeobjekte markiert und ihre Styles neu eingestellt werden.

Klicken Sie das Textobjekt mit der Zeile „Dieser Text IST kursiv.“ an. Der Style-Schaltfläche können Sie entnehmen, daß das Objekt mit dem Style „Normal“ verbunden ist. Klicken Sie auf die Schaltfläche „Style“, um sich die Liste der verfügbaren Styles anzeigen zu lassen, und doppelklicken Sie auf den neuen Style „Kursiv“, um ihn mit dem markierten Ausgabeobjekt zu verbinden.

Lassen Sie sich den Report (mit **DATEI | DRUCKBILD EINSEHEN**) anzeigen, um zu überprüfen, daß das Textausgabeobjekt „Dieser Text IST kursiv.“ auch wirklich mit dem neuen Style verbunden ist.

Bei dem soeben erstellten Report wird die Ausgabe des kursiven bzw. normalen Texts auf dem Bildschirm von Windows-Bildschirmtreibern gesteuert. Der Report läßt sich aber auch mit einer Non-Windows-Anwendung ausgeben, bei der keine Windows-Treiber verfügbar sind. Dazu müssen Sie druckerspezifische Steuersequenzen in den neuen Style integrieren.

Aktivieren Sie bei ausgewähltem Style „Kursiv“ die Menüoption **STYLE | DEFINITION FREMDFORMAT**, um die Dialogbox „Informationen über Fremdformat-(Non-Windows)Layout“ aufzurufen.

Der Epson-kompatible Code für Kursivdruck lautet (laut Angabe in den Druckerhandbüchern von Epson) „ESC+4“. Würden Sie diese Zeichenkette so in das Editierfeld „Einleitungssequenz“ eingeben und auf „OK“ klicken, würde CodeReporter eine Fehlermeldung anzeigen. Die einzigen unterstützten Formate für Steuersequenzen sind nämlich Dezimal- bzw. Hexadezimalwerte – und nicht die ASCII-Zeichen wie gedruckt.

Der Dezimalwert für die Escape-Taste ist 27. Die Vier wird als „52“ dargestellt (siehe Anhang D). Klicken Sie auf die Schaltfläche „Dez“, und geben Sie „27 52“ (ohne Anführungszeichen) in das Editierfeld „Einleitungssequenz“ ein.

Da der Kursivdruck nach der Ausgabe des Objekts zurückgesetzt werden muß (tun Sie das nicht, werden auch Objekte mit normaler Schrift kursiv ausgegeben), müssen Sie ihn auch wieder deaktivieren. Bei Epson-kompatiblen Druckern wird dazu „ESC+5“ verwendet. Geben Sie „27 53“ in das Editierfeld „Ausleitungssequenz“ ein, und klicken Sie auf „OK“, um die Non-Windows-Definition des Styles „Kursiv“ abzuschließen.

9 CodeReporter-Optionen

Das vorliegende Kapitel befaßt sich mit einigen der bei CodeReporter „oberflächlich“ definierbaren Einstellungen. Diese wirken sich in keiner Weise auf Reports oder Reportdateien aus, sondern verändern lediglich die Funktionsweise und das Erscheinungsbild von CodeReporter.

Ansichtsoptionen

Beim ersten Aufruf von CodeReporter haben die meisten Optionen noch ihre voreingestellten Werte. Die Ansichtsoptionen fallen dabei am ehesten ins Auge.

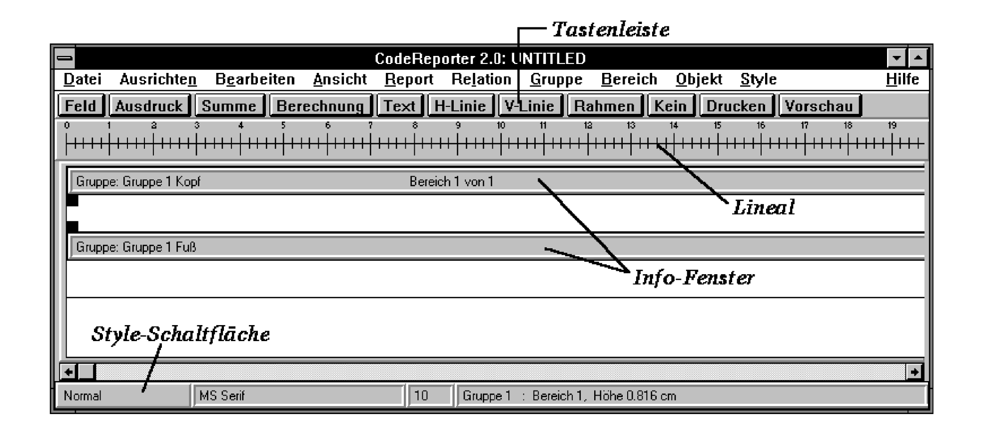


Abb. 9.1: Elemente im Report-Entwicklungsbildschirm

Die Anzeige bzw. unterdrückte Anzeige der verschiedenen Elemente (Tastenleiste, Lineal, Statusanzeige und Info-Fenster) im Report-Entwicklungsbildschirm wird über das Menü **ANSICHT** gesteuert.

Das Häkchen (Voreinstellung) neben den einzelnen Menüoptionen bedeutet, daß das entsprechende Element im Report-Entwicklungsbildschirm angezeigt wird.

Dabei können einige Elemente die Bildschirmfläche verkleinern, auf der der Report entwickelt wird.

Wählen Sie eine Option mit Häkchen aus, wird das Bildschirmelement entfernt, und das Häkchen verschwindet. Das Auswählen einer Option ohne Häkchen bewirkt das genaue Gegenteil.

Die so gegebene Möglichkeit, Elemente des Entwicklungsbildschirms ein- bzw. auszublenden, verleiht dem Reportentwickler ein Höchstmaß an Flexibilität bei der Gestaltung seiner Arbeitsumgebung.

Reportvorgaben

Über die Dialogbox „Reportvorgaben“ lassen sich weitere arbeitsspezifische Einstellungen vornehmen, als da sind: bevorzugte Maßeinheit, Maßeinheit für das Lineal (falls aktiviert) und Anzeigehöhe für den Report.

Maßeinheiten für die Anzeige

Intern legt CodeReporter die Positionen und Maße von Ausgabeobjekten und Reportbereichen in Schritten von 1/1 000 Zoll fest. Da in der Regel eine derartige Genauigkeit bei der Positionierung von Ausgabeobjekten nicht erforderlich ist, stellt CodeReporter für den Entwicklungsbildschirm austauschbare Maßeinheiten zur Verfügung.

Als kanadisches Programm ist CodeReporter auf „Zoll“ voreingestellt. Über die Dialogbox „Reportvorgaben“ läßt sich im Bereich „Maßeinheiten“ rasch eine andere Schaltfläche aktivieren. Auf diese Einheit wird dann bei jeder Operation mit Maßen zurückgegriffen. Dazu gehören das Format des Reports und der Reportbereiche, die Koordinaten und Maße von Ausgabeobjekten, die Ränder usw.

Automatisch werden alle Teile des Reports auf die neu eingestellte Maßeinheit umgestellt. Dabei verändert CodeReporter weder Position noch Größe der Reportelemente, da intern weiterhin alles in Schritten von 1/1 000 Zoll gemessen wird.

Auf die gemeinsamen Dialogboxen (z. B. „Schriftart“), die CodeReporter nicht verändern kann, und das Lineal wirkt sich die Einstellung der „Maßeinheiten“ nicht aus.

Das Lineal fungiert als optische Hilfe beim waagerechten Positionieren und Versetzen von Ausgabeobjekten. So könnte ein Report zum Beispiel auf spezielle Rechnungsformulare gedruckt werden, bei denen eine Spalte für bestimmte Zahlen vorgedruckt ist. Anstatt lange herumzuprobieren, kann man die tatsächlichen Maße in einem solchen Fall mit einem Lineal ermitteln und das Objekt dann rasch durch Übertragung der Maße auf das CodeReporter-Lineal plazieren.

Auch die Linealeinheiten werden in der Dialogbox „Reportvorgaben“ eingestellt. Die ausgewählte Linealeinheit, „Zoll“ oder „Zentimeter“, wird dann im Bildschirmlineal angezeigt. Diese Einstellung kann sich durchaus von der Einstellung der „Maßeinheit“ unterscheiden.

	Zoll	Punkt	Zentimeter
1 Zoll		72	2,54
1 Punkt	0,014		0,34
1 Zentimeter	0,41	29	

Tabelle 9.1: Umrechnungstabelle für Maßeinheiten

Seitengröße einstellen

Über die Menüoption **DATEI | DRUCKBILD EINSEHEN** können Sie sich vor dem Druckvorgang einen Report auf dem Bildschirm anzeigen lassen.

In den meisten Fällen sollen dabei einfach die Daten eines Reports eingesehen bzw. die Zurücksetzung der Seiten überprüft werden. Erleichtert wird das, indem CodeReporter das Seitenformat auf das Bildschirmformat einstellt. Dabei wird die Druckseite so weit verkleinert, daß sowohl Seitenkopf als auch Seitenfuß gleichzeitig zu sehen sind.

Deaktivieren Sie das Kontrollkästchen „Seitenformat entspricht beim Druckbildeinsehen Bildschirmformat“ in der Dialogbox „Reportvorgaben“, wird der Report eins zu eins angezeigt.

Schriftgrößen und Seitenformat stimmen in diesem Fall mit der späteren Druckseite überein. Da die meisten Monitore allerdings nicht in der Lage sind, eine volle Seite anzuzeigen, erscheinen senkrechte Bildlaufleisten, damit Sie sich auch den restlichen Inhalt der Seite ansehen können.

Hinweis: Das Kontrollkästchen „Seitenformat entspricht beim Druckbild-einsehen Bildschirmformat“ wirkt sich in keiner Weise auf den Ausdruck eines Reports aus, sondern einzig und allein auf die Bildschirmanzeige über die Menüoption **DATEI | DRUCKBILD EINSEHEN**.

10 Reports gestalten

Bei allen neuen Reports geht CodeReporter von bestimmten Annahmen aus. Parameter wie Seitenformat, Reportbreite, Ränder, Zahlenformat usw. werden bei der Erstellung eines neuen Reports auf die voreingestellten Werte zurückgesetzt.

Aber nicht immer stimmen diese Voreinstellungen mit den Vorstellungen des Reportentwicklers oder dem Ziel des Reports überein. Nicht alle Reports werden auf Papier derselben Größe gedruckt und haben auch keine gleich großen Ränder. Bei vielen Reports sind Anpassungen erforderlich, damit sie den an sie gerichteten Anforderungen gerecht werden.

Die genannten und auch andere Einstellungen können nach der Anlage eines Reports innerhalb der Dialogboxen „Ränder“ und „Reportvorgaben“ vorgenommen werden.

Ränder und Seitenformat

In der Dialogbox „Ränder“ (Abb. 10.1), die mittels der Menüoption **REPORT | RÄNDER** aufgerufen wird, können Sie sowohl die Ränder des Reports als auch die Seitenbreite einstellen.

Ränder und Seitenformat werden automatisch auf die maximal verfügbare Druckfläche des Standard-Windows-Druckers eingestellt. Bei den meisten Nadeldruckern bedeutet das, daß der Report sich anfangs über die ganze Seite erstreckt. Bei anderen Druckern, insbesondere bei Laserdruckern, gibt es einen hardwaremäßig eingestellten nicht bedruckbaren Bereich, der beim Drucken frei bleibt. CodeReporter ermittelt diesen Bereich automatisch und sorgt dafür, daß die Randeinstellungen für einen Report den nicht bedruckbaren Bereich des Standard-Druckers nicht unterschreiten.

Ränder

Die Randeinstellungen des Reports erfolgen über die Eingaben in die Dialogbox „Ränder“. Um den Rand oben, unten, links oder rechts neu einzustellen, ändern Sie einfach den Wert im jeweiligen Editierfeld.

Hinweis: Seitenkopf- und Seitenfuß-Bereiche werden zwischen dem oberen und dem unteren Rand ausgegeben. Ist der obere Rand zum Beispiel auf 2,5 cm eingestellt, hat der erste Seitenkopf-Bereich einen Abstand von 2,5 cm zur oberen Papierkante.

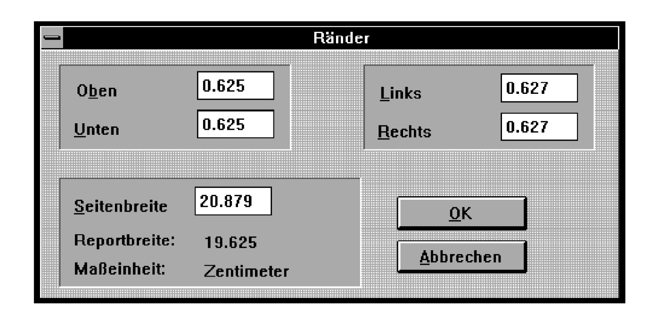


Abb. 10.1: Dialogbox „Ränder“

Hinweis: Um den Eindruck zu erwecken, daß Teile des Seitenkopfes bzw. -fußes außerhalb des oberen und unteren Randes eines Reports liegen, können Sie diese Ränder auf einen kleinen Wert einstellen und dann den Seitenkopf-/Seitenfuß-Bereich in der Höhe vergrößern.

Seitenbreite

Wie bereits erwähnt, richtet sich die Seitenbreite eines neuen Reports nach der Einstellung des Standard-Windows-Druckers. Soll die Endausgabe auf einem Drucker erfolgen, dessen Druckbreite größer bzw. kleiner als die des voreingestellten ist, können Sie die von CodeReporter verwendete „Seitenbreite“ entsprechend verändern.

Die Höhe der Seite wird während des Druckvorgangs über den ausgewählten Drucker ermittelt.

Achtung! Es liegt in der Verantwortung desjenigen, der den Report ausgibt, sicherzustellen, daß der Drucker für Windows richtig konfiguriert ist – dazu gehören auch die Überprüfung der Papiergröße und des Formats.

Papierformat

Ob ein Report im Hoch- bzw. Querformat gedruckt wird, entscheidet sich anhand der Druckerkonfiguration zum Zeitpunkt der Ausgabe. Die Einstellung erfolgt entweder über die Windows-Systemsteuerung oder aber temporär über die Menüoption **DATEI | DRUCKERAUSWAHL**. Einzelheiten zum Einstellen des Papierformats entnehmen Sie bitte dem Kapitel „Drucken“.

Reportvorgaben

Die Dialogbox „Reportvorgaben“ kann zur Abänderung report-weiter Voreinstellungen verwendet werden, die die Ausgabe des Reports beeinflussen – unter anderem die Formatierung von Zahlen und Daten.

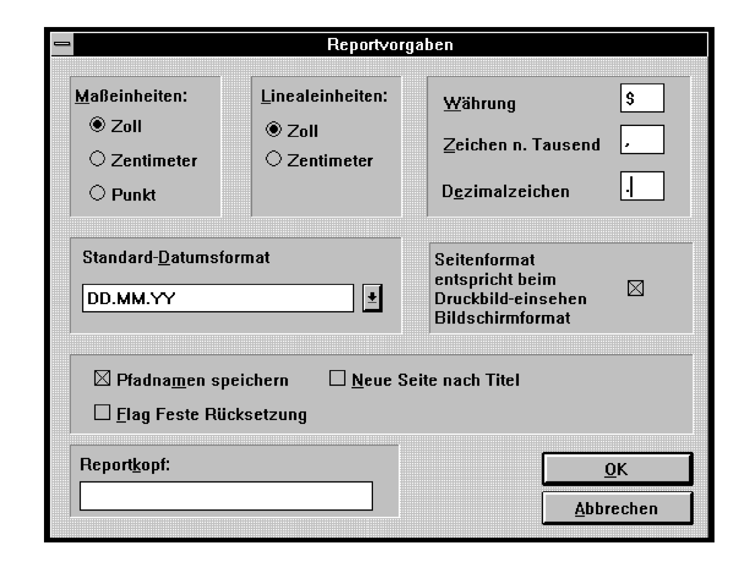


Abb. 10.2: Dialogbox „Reportvorgaben“

Zahlenformat

Ein Großteil der Einstellungen, die sich auf Ausgabeobjekte mit Zahlenwerten auswirken, erfolgen objektbezogen über die jeweilige Dialogbox „Objektdefinition“. Zu den Einstellungen gehören: Anzahl der angezeigten Dezimalstellen, ob

es sich bei dem Objekt um einen Prozent-, einen Währungswert oder eine ganze Zahl handelt, wie ein Nullwert ausgegeben wird usw.

Bei manchen numerischen Ausgabeobjekten sind die Einstellungen von Report zu Report verschieden. Dazu zählen die Verwendung des Zeichens nach Tausend, das im Report benutzte Währungssymbol, das Zeichen für das Dezimalkomma usw. In der Dialogbox „Reportvorgaben“ werden diese Einstellungen für den aktuellen Report vorgenommen.

Sollen die numerischen Ausgabeobjekte als Währungswerte ausgegeben werden, wird/werden das/die im Editierfeld „Währung“ angegebene(n) Zeichen dem Inhalt des Objekts unmittelbar vorangestellt. (Einzelheiten zur Formatierung von Ausgabeobjekten als Währungswerte finden Sie im Abschnitt „Zahlen“ im Kapitel „Ausgabeobjekte“.) Das voreingestellte Währungssymbol ist das Dollarzeichen ($); Sie können hier jedoch bis zu zehn Zeichen eingeben.

Tabelle 10.1 enthält einige der üblichen Währungssymbole, die der Standard-Windows-Zeichensatz unterstützt.

Zeichen	Name	Tastenkombination
¢	Cent	Alt+162
£	Pfund	Alt+163
¥	Yen	Alt+165

Tabelle 10.1: Verbreitete Währungssymbole

Zahlenwerte ab Tausend lassen sich mit Hilfe des Tausendertrennzeichens in Dreiergruppen gliedern, Tausend, Million, Milliarde usw. In Nordamerika wird dazu das Komma verwendet – die CodeReporter-Voreinstellung.

In das Editierfeld „Zeichen n. Tausend“ können Sie ein beliebiges Zeichen (auch ein Leerzeichen) für diesen Wert eingeben oder es allzumal löschen.

Haben Sie ein Tausendertrennzeichen angegeben, wird es bei allen Zahlenwerten verwendet. Wird das Zeichen gelöscht, erscheinen die Zahlen unformatiert.

Beispiel:

Mit Zeichen nach Tausend	1.000.000
Ohne Zeichen nach Tausend	1000000

Bei der Ausgabe von Dezimalbrüchen setzt CodeReporter das im Editierfeld „Dezimalzeichen“ angegebene Zeichen zwischen die ganze Zahl und den Bruchteil. Auch in dieses Editierfeld können Sie jedes beliebige Zeichen eingeben. In Nordamerika wird dazu der Dezimalpunkt verwendet, in Deutschland und anderen europäischen Staaten das Komma.

Hinweis: Diese Einstellung wirkt sich nicht auf die Anzahl der bei einem Objekt ausgegebenen Dezimalstellen aus. (Wie man die Anzahl der Dezimalstellen festlegt, entnehmen Sie bitte dem Abschnitt „Zahlen“ im Kapitel „Ausgabeobjekte“.)

Datumsformat

Ausgabeobjekte, die als Ergebnis einen Datumswert haben (z. B. Datumsfelder, Datumsausdrücke), lassen sich in verschiedenen Formaten ausgeben. Bei neuen Ausgabeobjekten gilt bis auf weiteres die Einstellung der einzeiligen Listbox „Standard-Datumsformat“.

In den editierbaren Teil des Feldes können Sie eine bestimmte Datumsschablone eingeben oder ein vordefiniertes Datumsformat in der Liste markieren und auswählen.

Einzelheiten zu Datumsschablonen entnehmen Sie bitte dem Abschnitt „Datum“ im Kapitel „Ausgabeobjekte“.

Hinweis: Eine Änderung des Wertes in der Listbox „Standard-Datumsformat“ wirkt sich nicht auf das Datumsformat bereits vorhandener Datumsausgabeobjekte aus. Die Einstellung von „Standard-Datumsformat“ bezieht sich ausschließlich auf neu zu erstellende Objekte.

Pfadnamen

In einem Report kann CodeReporter mehrere Datendateien verwenden, die sich auf verschiedenen Laufwerken und in verschiedenen Verzeichnissen befinden. Zu diesem Zweck werden innerhalb des Reports die vollständigen Pfadnamen ge-

speichert. Mögliche Datenquellen lassen sich auf diese Weise zwar flexibel angeben, doch dürfen die Datendateien in dem Verzeichnis, in dem sie sich bei der Erstellung des Reports befunden haben, nicht fehlen.

Das Kontrollkästchen „Pfadnamen speichern“ in der Dialogbox „Reportvorgaben“ bestimmt, ob die vollständigen Pfadnamen der Datendateien zusammen mit dem Report gespeichert werden oder nicht.

Ist das Kontrollkästchen nicht aktiv, sichert CodeReporter Laufwerk und Verzeichnis der Datendatei nicht mit. Beim nächsten Laden des Reports kann CodeReporter nicht auf die Ausgangspfade der Datendateien zugreifen (da sie nicht gespeichert wurden) und sucht sie im aktuellen Verzeichnis.

Feste Zurücksetzung

Tritt eine Gruppenrücksetzbedingung für eine Gruppe ein und die Option „Seite zurücksetzen“ ist eingestellt (siehe Kapitel „Gruppen“), wird anhand des Kontrollkästchens „Flag Feste Rücksetzung“ die Zurücksetzungsmethode für die Seite ermittelt.

Wenn das „Flag Feste Rücksetzung“ nicht aktiviert ist (Voreinstellung), wird keine neue Seite generiert, falls die Gruppe mit der Option „Seite zurücksetzen“ aufgrund einer Gruppenrücksetzung auf höherer Ebene zurückgesetzt wurde. Eine neue Seite wird lediglich dann generiert, wenn der Gruppenausdruck der Gruppe selbst für die Gruppenrücksetzbedingung verantwortlich ist. Ist das Kontrollkästchen „Flag Feste Rücksetzung“ aktiv, wird desungeachtet eine neue Seite generiert.

Abb. 10.3 verdeutlicht die Funktionsweise beider Einstellungen.

Seitenumbruch nach dem Titel

Über das Kontrollkästchen „Neue Seite nach Titel“ bestimmen Sie, ob CodeReporter nach der Ausgabe des Titelbereichs eine neue Seite beginnen soll oder nicht.

Ist das Kontrollkästchen markiert, wird der Titelbereich des Reports, falls vorhanden, ausgegeben und der Report auf der nächsten Seite fortgesetzt. Wenn das Kontrollkästchen nicht markiert ist, erfolgt die Ausgabe der Titelgruppe auf derselben Seite wie die des Kopfbereichs der ersten Gruppe.

Reportkopf

Per Voreinstellung wird beim Druckbild-einsehen „CodeReporter 2.0“ in der Titelleiste angezeigt. Dieser Text läßt sich auf den Namen des Reports bzw. auf jede andere sinnvolle Bezeichnung einstellen. Bearbeiten Sie dazu das Editierfeld „Reportkopf“ in der Dialogbox „Reportvorgaben“.

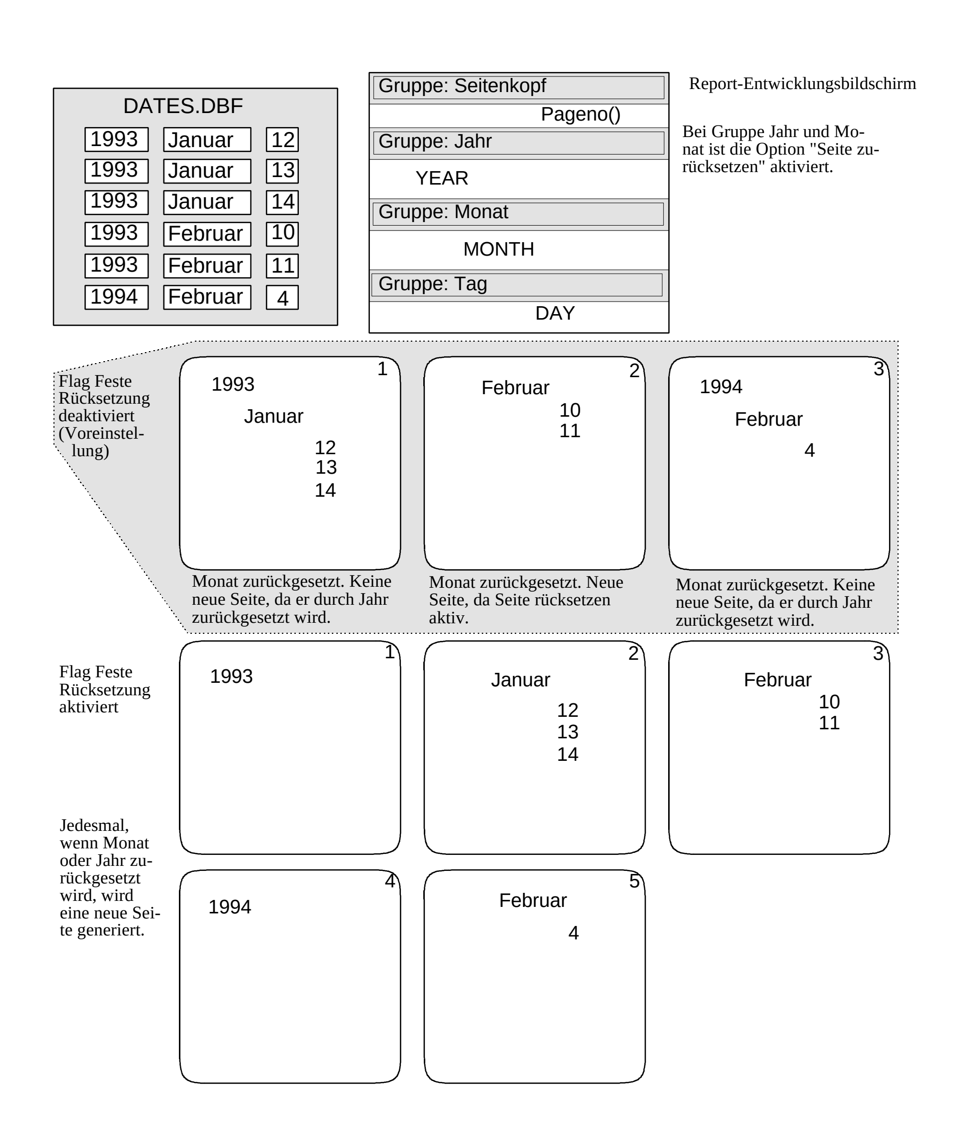


Abb. 10.3: Seite zurücksetzen und Flag Feste Rücksetzung

11 Drucken

Jeder Report soll am Ende auf Papier ausgegeben werden, das Ausgabegerät ist gewöhnlich ein Drucker. Es gibt aber auch Situationen, in denen ein Report besser auf dem Bildschirm ausgegeben bzw. in eine Datei geschrieben wird.

Das vorliegende Kapitel erläutert die bei der Reportausgabe erforderlichen Operationen.

Einen Drucker auswählen

Viele Hardware-Konfigurationen setzen sich aus einem Computer und einem daran angeschlossenen Drucker zusammen, über den alle Druckausgaben erfolgen. In diesem Fall kann man nur einen Drucker auswählen.

Bei anderen Konfigurationen werden mehrere Drucker lokal und/oder über ein Netzwerk an einen Computer angeschlossen. Solange Sie CodeReporter keine andersgearteten Anweisungen geben, wird für alle Druckausgaben der Windows-Standarddrucker benutzt.

Handelt es sich hierbei um einen geeigneten Drucker, brauchen Sie die Einstellung nicht zu verändern. Jeder andere Drucker muß in der Dialogbox „Druckereinrichtung" explizit ausgewählt werden (Abb. 11.1). Die Dialogbox wird über die Menüoption **DRUCKEN | DRUCKERAUSWAHL** oder die Schaltfläche „Einrichten" in der Dialogbox „Drucken" (Abb. 11.2) aufgerufen.

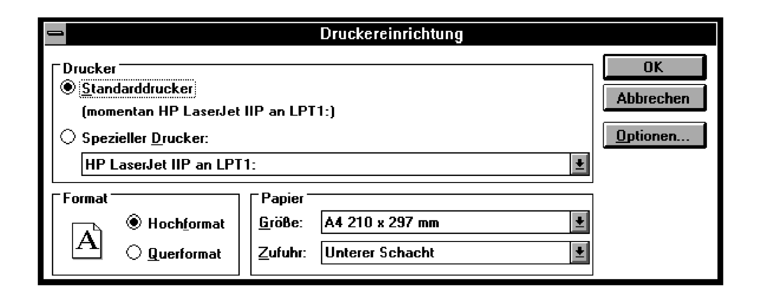


Abb. 11.1: Dialogbox „Druckereinrichtung"

Einen nicht voreingestellten Drucker wählen Sie über die Optionsschaltfläche „Spezieller Drucker“ aus der daraufhin angezeigten Listbox aus.

In dieser Dialogbox geben Sie ebenfalls an, ob Sie im Hoch- oder Querformat drucken wollen, welches Papierformat Sie wünschen usw. Wählen Sie die für den Report erforderlichen Optionen an, und klicken Sie auf „OK“.

Achtung! Die Einstellungen der Dialogboxen „Drucken“ sowie „Druckereinrichtung“ werden nicht zusammen mit dem Report gespeichert und müssen deshalb beim Laden des entsprechenden Reports jedesmal neu vorgenommen werden.

Auf den Bildschirm

Mit CodeReporter können Sie sich einen Report ansehen, ehe Sie ihn auf einem Drucker ausgeben. Dabei können Sie anhand der Bildschirmanzeige überprüfen, ob das Layout des Reports so in Ordnung ist – ohne Papier zu verbrauchen.

Die Menüoption **DATEI | DRUCKBILD EINSEHEN** öffnet ein neues Fenster und gibt den Report seitenweise darin aus. CodeReporter gibt das Fenster als Vollbild (über den ganzen Bildschirm) aus. Mittels der Schaltflächen „Symbol“ bzw. „Vollbild“ kann dieses Fenster verkleinert oder vergrößert werden, und über das Systemmenü läßt sich seine Größe individuell verändern.

Mit der Menüoption **NÄCHSTE** springen Sie auf die nächste Seite, während Sie das Fenster mit **SCHLIESSEN** schließen und zum CodeReporter-Entwicklungsbildschirm zurückkehren.

Hinweis: Beim Einsehen eines Reports kann die Druckseite vorübergehend auf Bildschirmgröße reduziert werden. Sie können sich aber den Report von vornherein auch im Seitenformat des aktuellen Druckers anzeigen lassen. Einzelheiten dazu entnehmen Sie bitte dem Abschnitt „Seitengröße einstellen“ im Kapitel „CodeReporter-Optionen“.

Auf einen Drucker

Nachdem Sie sich den Report auf dem Bildschirm kontrolliert und für gut befunden haben, wird es Zeit, ihn mittels der Menüoption **DATEI | DRUCKEN** auf dem ausgewählten Drucker auszugeben. Die Menüoption ruft die gemeinsame Dialogbox „Drucken“ auf (Abb. 11.2).

Geben Sie die gewünschte Anzahl der Kopien ein, und/oder klicken Sie auf „Einrichten“, um den gewünschten Drucker auszuwählen und einzurichten. Klikken Sie auf „OK“, wird der Druckvorgang gestartet; ein Klick auf „Abbrechen“ bewirkt den Rücksprung in den CodeReporter-Entwicklungsbildschirm, ohne zu drucken.

Hinweis: Bei einem Report ohne Seitenumbrüche macht es keinen Sinn, eine bestimmte Anzahl von Seiten ausgeben zu lassen. Alle hier eingegebenen Werte werden von CodeReporter ignoriert.

Stellen Sie die einzeilige Listbox „Druckqualität“ auf „Entwurf“ ein, kann es geschehen, daß der Druckertreiber den waage- bzw. senkrechten Abstand von Ausgabeobjekten bis an die Zeichengrenze verschiebt.

Das kann dazu führen, daß Reports, die beim Einsehen noch einzeilig waren, mit jeweils einer Leerzeile zwischen den Zeilen ausgegeben werden. Stellen Sie also „Druckqualität“ auf „Brief“ ein, oder setzen Sie die Größe der Reportbereiche auf 12 Punkt (oder ein Vielfaches davon).

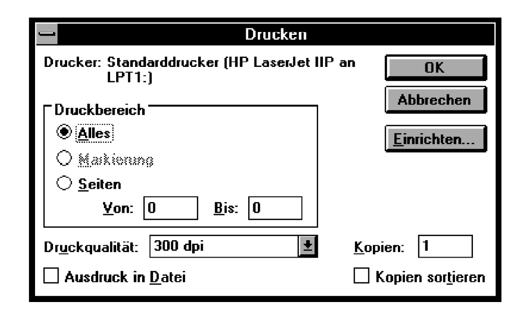


Abb. 11.2: Dialogbox „Drucken“

In eine Datei

Aus verschiedenen Gründen sollte die gedruckte Version eines Reports auch als Datei vorliegen. Zum einen kann er dann später noch einmal ausgedruckt werden, zum anderen kann man ihn in ein anderes Programm übertragen, und zum dritten läßt er sich auf diese einfache Weise elektronisch speichern.

Folgende Schritte müssen Sie unternehmen, um einen Report in eine Datei zu drucken:

1. Rufen Sie über die Menüoption **DATEI | DRUCKEN** die Dialogbox „Drucken“ auf.
2. Aktivieren Sie das Kontrollkästchen „Ausdruck in Datei“.
3. Klicken Sie auf „OK“, um den Druckvorgang zu starten.
4. Geben Sie bei Erscheinen der Dialogbox „Ausdruck in Datei“ den Dateinamen (und Pfad) ein, unter dem der gedruckte Report abgelegt werden soll. Dieser Name sollte sich deutlich von dem der Reportdatei unterscheiden (heißt der aktuelle Report z. B. RECHNUNG.REP, sollte die Druckdatei eben nicht genauso heißen).

Drucken kontra Einsehen

Eine Reportdruckdatei enthält alle Angaben, um einen Report in der gewünschten Schriftart, -größe, Formatierung usw. ausgeben zu können. Wie im Abschnitt „Non-Windows-Styles“ erwähnt, kann Windows über seine Druckertreiber mit nahezu jedem Drucker zusammenarbeiten.

Beim Drucken in eine Datei schreibt Windows alle druckerspezifischen Steuerzeichen für Linien, Grafiken, Schriften usw. ebenfalls in die Datei. Wird diese nun später auf den Drucker kopiert, sieht der Report genauso aus, als wäre er unmittelbar aus der Anwendung gedruckt worden.

Das Erstellen einer Reportdruckdatei bringt allerdings zwei nicht zu unterschätzende Nachteile mit sich:

- Zum einen kann die Datei recht groß werden – besonders dann, wenn es sich um einen umfangreichen Report mit mehreren Schriftarten, Grafiken usw. handelt.
- Zum anderen ist die Datei, da sie alle Druckersteuerzeichen enthält, für andere Anwendungen (z. B. Textverarbeitungsprogramme) nicht lesbar.

Um nicht alle druckerspezifischen Daten in die Datei übernehmen zu müssen, verfügt Windows über einen speziellen Universaldruckertreiber, der lediglich Text ausgibt. Grafische Elemente (Linien, Rahmen, Grafiken) und Schriften werden nicht mit ausgegeben. Erfolgt das Drucken in eine Datei mit diesem Treiber, wird die Datei nicht übermäßig groß, und sie kann auch mit anderen Anwendungen verarbeitet werden.

Der Universaldruckertreiber ist ein Bestandteil von Windows und muß installiert sein, damit CodeReporter ihn benutzen kann. Die Installation erfolgt über die Windows-Systemsteuerung, und zwar die Druckereinstellung.

- Klicken Sie in der Dialogbox „Drucker“ auf die Schaltfläche „Drucker hinzufügen“.
- Wählen Sie in der Listbox „Druckerliste“ den Drucker „Universal/Nur Text“ aus, und
- klicken Sie auf die Schaltfläche „Installieren“.

Danach können Sie diesen neuen Drucker auswählen und Ihre Reports auf reine Textdrucker oder als ASCII-Dateien ausgeben lassen.

Hinweis: Wenn Sie einen Report mit dem Universaldruckertreiber in eine ASCII-Datei schreiben, sollte die Höhe der Reportbereiche auf 12 Punkt (oder ein Vielfaches davon) eingestellt sein, damit der Report mit dem richtigen Zeilenabstand gespeichert wird.

In eine Datenbankdatei

Ein wichtiges Leistungsmerkmal von CodeReporter ist seine Fähigkeit, einen Report in eine Datendatei auszugeben. Dabei werden alle Ergebnisse, u. a. Summen, Berechnungen, Felder, in der Datendatei abgelegt, die dann als Grundlage für einen weiteren Report dienen kann.

Um das zu bewerkstelligen, werden die Ausgabeobjekte angegeben, die in die Ausgabedatendatei übernommen werden sollen. Da die meisten Ausgabeobjekte jedoch dynamische Objekte sind, deren Wert sich von zusammengesetztem Datensatz zu zusammengesetztem Datensatz ändert, müssen Sie auch angeben, wann die Werte der Ausgabeobjekte in die Ausgabedatendatei geschrieben werden sollen.

Diese beiden Aufgaben werden mit Hilfe der Dialogbox „Schablone für Ausgabedatei“ bewältigt, das über die Menüoption **REPORT | VORGABEN AUSGABEDATEI** aufgerufen wird. Die in dieser Dialogbox vorgenommenen Einstellungen werden zusammen mit dem Report gespeichert und können jederzeit verändert werden.

Objekte

In der Listbox „Objekte“ werden alle Ausgabeobjekte des Reports aufgeführt – bis auf Memofelder, Linien, Rahmen, Grafiken und Objekte im Titel-/Schlußbereich und in den Seitenkopf-/-fuß-Bereichen.

Wählen Sie ein Objekt an, werden seine entsprechenden Daten im Fenster „Objektinformationen“ angezeigt. Um Objekte zur Ausgabedatendatei hinzuzufügen, brauchen Sie sie lediglich zu markieren und auf die Schaltfläche „Hinzufügen“ bzw. „Alle hinzufügen“ zu klicken.

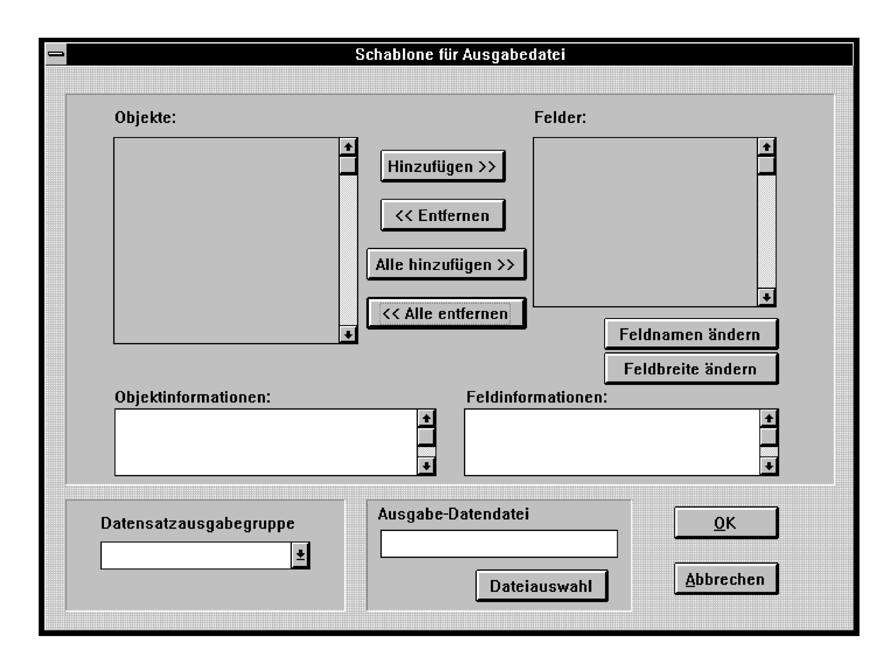


Abb. 11.3: Dialogbox „Schablone für Ausgabedatei“

In Fällen, in denen es sich bei der Bezeichnung des Objekts nicht um einen gültigen Feldnamen handelt (z. B. bei Summen und Berechnungen) oder falls in der Ausgabedatendatei bereits ein gleichnamiges Feld vorhanden ist, bittet Code-Reporter um die Eingabe eines neuen Feldnamens.

Nach dem Hinzufügen können die Feldangaben über die Schaltflächen „Feldnamen ändern“ und „Feldbreite ändern“ neu definiert werden.

Datensatzausgabegruppe

Mittels des Rücksetzausdrucks der „Datensatzausgabegruppe“ bestimmt CodeReporter den Zeitpunkt, an dem der Inhalt der Feldausgabeobjekte in einen neuen Datensatz geschrieben wird. Tritt für die angegebene Gruppe eine Rücksetzbedingung ein, wird auf der Platte ein neuer Datensatz angelegt und die Werte des Ausgabeobjekts in diesen Datensatz geschrieben; die verwendeten Werte stammen aus dem letzten Datensatz der Untermenge der vorhergehenden Gruppe (d. h., dem Gruppenfuß).

Per Voreinstellung enthält die einzeilige Listbox „Datensatzausgabegruppe“ die am weitesten außen gelegene Gruppe des Reports. Wünschen Sie eine andere Gruppe, kann sie über diese Listbox ausgewählt werden.

Ausgabe-Datendatei

Bei der „Ausgabe-Datendatei“ handelt es sich um die Datendatei, in der der Report abgelegt werden soll. Um den Dateinamen einzugeben, klicken Sie auf „Dateiauswahl“, damit die Dialogbox „Datendatei auswählen“ (eine gemeinsame Dialogbox zum Öffnen einer Datei) aufgerufen wird. Hier geben Sie Laufwerk, Verzeichnis und Namen der gewünschten Ausgabedatendatei ein.

Ist die angegebene Datendatei bereits vorhanden, gibt CodeReporter vor dem Überschreiben eine Meldung aus.

Wenn Sie auf „OK“ klicken, wird die neue Datendateischablone innerhalb des aktuellen Reports gespeichert.

In eine Datendatei drucken

Nach dem Erstellen der Datendateischablone kann der Report über die Menüoption **DATEI | IN DATENDATEI DRUCKEN** ausgegeben werden. Dabei wird die in der Dialogbox „Schablone für Ausgabedatei“ angegebene Datendatei angelegt und mit den Daten aus dem Report gefüllt. Mit der neuen Datendatei läßt sich danach arbeiten wie mit jeder anderen Datendatei auch.

Beispiel

Wie zweckmäßig es sein kann, einen Report in eine Datendatei zu drucken, soll das vorliegende kurze Beispiel verdeutlichen. Dabei wird die Datendatei PERSONEL.DBF sortiert und in die Datei PERSONS.DBF umgewandelt.

Aktivieren Sie nach dem Aufruf von CodeReporter die Menüoption **DATEI | NEU**, um einen neuen Report zu beginnen, und machen Sie die Datendatei PERSONEL.DBF zum Top Master.

Fügen Sie über die Schaltfläche „Feld“ in der Tastenleiste alle Felder der Datei in den Report ein, und klicken Sie auf die Gruppe „Body“. Da dieser Report in erster Linie zur Übertragung von Daten verwendet wird, kommt es auf die Plazierung der Felder in dem Bereich nicht an.

In der Datei PERSONEL.DBF liegen die Namen jeweils getrennt in Vor- und Nachnamenfeldern vor (FNAME bzw. LNAME). In der neuen Datendatei soll für die Namen jeweils nur ein Feld im Format „Smith, John“ vorhanden sein.

Mit CodeReporter bereitet das keine Schwierigkeiten. Löschen Sie einfach die beiden Namensfelder, und fügen Sie ein Ausdrucksausgabeobjekt ein, das beide Felder enthält. Zum Beispiel:

```
TRIM( LNAME ) + ', ' + FNAME
```

Wenn Sie sich den Report jetzt anzeigen lassen, enthält er zwar wie vorher alle Daten der Datei PERSONEL.DBF, doch nur ein kombiniertes Namensfeld.

Da der Report nun alle Felder enthält, kann die Schablone für die Ausgabedatei generiert werden. Wählen Sie die Menüoption **REPORT | VORGABEN AUSGABEDATEI**, und rufen Sie die Dialogbox „Schablone für Ausgabedatei“ auf. Klicken Sie auf die Schaltfläche „Alle hinzufügen“, um alle Ausgabeobjekte des Reports in die neue Datendatei zu übernehmen.

Während das geschieht, meldet CodeReporter, daß dem Ausdrucksausgabeobjekt ein Name für die neue Datendatei fehlt. Geben Sie „NAME“ in das Editierfeld „Neuer Name“ ein, und klicken Sie auf „OK“.

Klicken Sie auf die Schaltfläche „Dateiauswahl“, und geben Sie in der Dialogbox als Bezeichnung für die neue Datendatei „PERSONS.DBF“ ein. Da alle anderen Einstellungen der Dialogbox „Schablone für Ausgabedatei“ in Ordnung sind, klicken Sie auf „OK“, um die Schablone zu speichern.

Falls erforderlich, können Sie über die Menüoption **REPORT | SORTIERAUSDRUCK** einen entsprechenden Ausdruck für den Report eingeben. Die Datensätze werden dann in dieser Reihenfolge in die Ausgabedatei geschrieben. Geeignete Sortierausdrücke sind `LNAME+FNAME`, `EMPID` oder `SALARY`.
Die neue Datendatei PERSONS.DBF wird über die Auswahl der Menüoption **DATEI | IN DATENDATEI DRUCKEN** erzeugt.

Laden Sie die Datei PERSONS.DBF mit **DATEI | NEU** (und speichern die aktuelle Datei vorher ab), um zu überprüfen, ob die Daten wunschgemäß übernommen worden sind.

12 Funktionsübersicht

Hinweis: Im vorliegenden Kapitel erläutern wir die Funktionen und Techniken für den Einsatz des Reportmoduls in C/C++. CodeBasic-Programmierer lesen bitte Anhang F.

Die Funktionen des Reportmoduls sind dazu da, maßgeschneiderte Reports zu entwickeln. Mit dem Reportmodul ist es ein leichtes, Daten aus Daten-, Index- und Memodateien zusammenzufassen, zu formatieren, anzuzeigen und auszudrucken.

Auf alle Funktionen des Reportmoduls können Sie indirekt über CodeReporter zugreifen. CodeReporter verwendet das Reportmodul zur Erzeugung variabel codierter Reportdateien und deren Quellcodes in C. Der generierte Quellcode enthält zahlreiche Aufrufe der Funktionen des Reportmoduls.

In einigen Fällen ist es möglicherweise erforderlich, einen Report vollständig per Hand anzulegen oder einen mit CodeReporter erstellten Report zu bearbeiten. Unter Verwendung des Reportsmoduls kann die Anwendung einen CodeReporter-Report laden, Abfrage- und/oder Sortierausdrücke verändern, das Ausgabegerät bestimmen und den Report schließlich ausführen. Zur Durchführung dieser bestimmten Aufgaben werden unmittelbar die Reportfunktionen eingesetzt.

Hinweis: Das Reportmodul enthält keine Funktion, um eine Berechnung zu erstellen, denn **expr4calc_create** ist Bestandteil des Ausdrucksauswertungsmoduls von CodeBase 5. Einzelheiten hierzu finden Sie im CodeBase-Referenzhandbuch sowie unter **obj4calcCreate**.

Bezeichnungen des Reportmoduls

Die Funktionen des Reportmoduls gliedern sich in Gruppen mit ähnlicher Funktionsweise, z. B. Funktionen, die sich auf den gesamten Report auswirken, Funktionen, die Gruppen erzeugen und verarbeiten, usw. In gewissem Sinne besteht das „Reportmodul“ aus einer Vielzahl von Einzelmodulen.

Jede Gruppe kann einen oder mehrere Bereiche umfassen, in die Ausgabeobjekte eingefügt werden können. Die Funktionen, die die Bereichsgröße verändern und die Unterdrückungsbedingungen bestimmen, beginnen mit **area4**.

Jeder Report enthält eine Anzahl von Gruppen. Die Funktionen, die sich auf das Verhalten der Gruppe und auf den Zugriff darauf auswirken, beginnen mit **group4**.

Die Funktionen, mit denen Ausgabeobjekte erzeugt, gelöscht und modifiziert werden, beginnen mit **obj4**.

Die sogenannten reportspezifischen Funktionen beginnen mit **report4** und werden eingesetzt, wenn etwas für den gesamten Report gilt. Die Funktionen initialisieren den Report, laden und speichern Relationen und Styles, verändern das Seitenformat, bearbeiten die Voreinstellungen des Reports usw.

Jedem Ausgabeobjekt lassen sich Ausgabemerkmale wie Schriftart und Farbe zuweisen. Diese Styles müssen in einem Report lediglich einmal, also nicht für jedes einzelne Ausgabeobjekt, definiert werden. Mit den **style4**-Funktionen werden die Styles eines Reports erzeugt und bearbeitet.

Die Definitionen, auf denen die Summenausgabeobjekte basieren, werden mit den **total4**-Funktionen erstellt und gelöscht. Die Summenausgabeobjekte selbst werden aber nicht mit diesen, sondern mit den **obj4**-Funktionen erzeugt.

Im folgenden Abschnitt erläutern wir alle oben genannten „Reportmodule“ im Detail.

Hinweis: Jede Anwendung kann gleichzeitig nur einen Report laden. Soll eine Anwendung mit mehreren Reports arbeiten, muß zwischen den einzelnen Reports die Funktion **report4free** aufgerufen werden.

Als Code speichern

Wenn CodeReporter einen Report über die Menüoption **DATEI | SPEICHERN** auf Platte sichert, wird dabei eine variabel codierte Reportdatei mit der Erweiterung „.REP“ erzeugt. Das reicht so, auch für Endbenutzer-Anwendungen, meistens aus, da die Reportdatei in die laufende Anwendung geladen werden kann. Auf

diese Weise kann der Entwickler einen Reportaufbau modifizieren, ohne gleich die Anwendung neu kompilieren zu müssen, die sich des Reports bedient.

In einigen Fällen ist es jedoch wünschenswert, den Report unmittelbar als Quellcode abzulegen. Das geschieht über die Menüoption **DATEI | SPEICHERN ALS CODE**, die die Dialogbox „Reportdatei zum Speichern angeben“ aufruft, in der der Dateiname und die Programmiersprache für den Code eingegeben werden.

CodeReporter erzeugt programmiersprachenspezifischen Quellcode für den aktuellen Report und speichert ihn in der angegebenen Datei ab. Die Quellcodedatei enthält zwei Funktionen, die in einer Anwendung aufgerufen werden können, um den Report und/oder die Relation zu laden. Diese beiden Funktionen müssen in der Anwendung vor ihrem Einsatz als Prototypen vorliegen, können aber in der generierten Quelldatei (und dem Prototypen) beliebig umbenannt werden.

buildRelate

Aufruf RELATE4 *buildRelate(CODE4 *cb, int openFiles)

Beschreibung Diese Funktion erzeugt und/oder bestückt eine **RELATE4**-Struktur für die Relation des gespeicherten Reports. Diese Struktur kann zusammen mit dem Relationsmodul von CodeBase 5 oder mit der Reportmodulfunktion **report4init** eingesetzt werden.

Sie wird automatisch von **buildReport** aufgerufen.

Parameter

cb	Ein Zeiger auf die **CODE4**-Struktur der Anwendung. Er dient der Speicherverwaltung und der Fehlerbehandlung.
openFiles	Dieser Parameter bestimmt, ob **buildRelate** automatisch die in der Reportdatei genannten Datendateien öffnen soll. Ist *openFiles* auf wahr gesetzt (nicht-Null), werden die Datendateien geöffnet, falls sie noch nicht geöffnet sind. Ist *openFiles* falsch (Null), wird davon ausgegangen, daß die Datendateien bereits geöffnet sind.

Rückgaben Bei Erfolg gibt diese Funktion einen gültigen Zeiger auf eine **RELATE4**-Struktur zurück. Konnten die Datendateien für die Relation nicht gefunden werden, wird Null zurückgegeben.

buildReport

Aufruf REPORT4 *buildReport(CODE4 *cb, int openFiles)

Beschreibung Diese Funktion baut den Report auf und gibt einen Zeiger auf die bestückte Reportstruktur zurück.

buildReport ruft automatisch **buildRelate** auf, um die hinter dem Report stehende Relation aufzubauen.

Parameter

cb	Ein Zeiger auf die **CODE4**-Struktur der Anwendung. Er dient der Speicherverwaltung und der Fehlerbehandlung.
openFiles	Dieser Parameter bestimmt, ob **buildRelate** automatisch die in der Reportdatei genannten Datendateien öffnen soll. Ist *openFiles* auf wahr gesetzt (nicht-Null), werden die Datendateien geöffnet, falls sie noch nicht geöffnet sind. Ist *openFiles* falsch (Null), wird davon ausgegangen, daß die Datendateien bereits geöffnet sind.

Rückgaben Bei Erfolg gibt diese Funktion einen gültigen Zeiger auf eine **REPORT4**-Struktur zurück. Konnten die Datendateien für die Relation nicht gefunden werden oder reichte der Speicher für den Aufbau des Reports nicht aus, wird Null zurückgegeben.

Reportfunktionen einsetzen

Mit Hilfe der Reportfunktionen können Sie innerhalb und außerhalb von Windows komplexe Reports entwickeln. In den meisten Fällen ist es eine mühselige

Angelegenheit, einen Report von Grund auf per Hand zu programmieren. Als Lösungsmöglichkeit dafür bietet CodeReporter dem Entwickler zwei Optionen an: entweder einen Report als variabel codierte Reportdatei (Erweiterung .REP) abzulegen oder den Report als C-Quellcode zu speichern.

Ein mit CodeReporter gespeicherter Report wird in einer Reportdatei abgelegt, die unmittelbar in eine Anwendung geladen werden kann. Mit nur wenigen Funktionsaufrufen läßt sich ein Report laden und rasch ausführen. In den meisten Fällen reicht das völlig aus.

In den Fällen, in denen Reportdateien sich als ungeeignet erweisen, kann man den Report mit CodeReporter als C-Quellcode speichern. Die Programmzeilen lassen sich zeitsparend einsetzen, da die für die Erzeugung des Reports erforderlichen Funktionen nicht einzeln programmiert werden müssen.

Es gibt einige wenige Fälle, in denen weder eine variabel codierte Reportdatei noch der von CodeReporter generierte Quellcode die Anforderungen an die Reporterstellung befriedigen. Die Reportmodulfunktionen von CodeReporter sind dazu da, einen Report zu modifizieren bzw. ganz von vorn aufzubauen, um den Anforderungen der Anwendung zu entsprechen.

Eine Reportdatei verwenden

Um einen Report in einer Anwendung zu implementieren, wird üblicherweise eine variabel codierte Reportdatei geladen und ausgeführt. Das Programm REP1.C verwendet eine CodeReporter-Reportdatei.

```
#include "d4all.h"

#ifdef __TURBOC__
   extern unsigned _stklen = 10000 ;
#endif

void main( int argc, char *argv[] )
{
   CODE4  cb ;
   REPORT4 *report ;
   if( argc < 2 )
      return ;

   d4init( &cb ) ;

   report = report4retrieve( &cb, argv[1], 1, NULL) ;
```

```
   if( report )
   {
      report4do( report ) ;
      report4free( report, 1, 1) ;
   }

   d4init_undo( &cb ) ;
   return ;
}
```

Die Funktion **report4retrieve** lädt die angegebene Reportdatei von der Platte, erzeugt eine interne **REPORT4**-Struktur und gibt einen Zeiger auf die Struktur zurück. Dieser Zeiger wird verwendet, um den Report mit der Funktion **report4do** auszuführen.

Mit dem Code in REP1.C läßt sich jede beliebige Reportdatei ausführen, da ihr Name als Befehlszeilenparameter übergeben wird. Um einen Report von der Platte zu laden, ist lediglich der Dateiname erforderlich. Der gesamte Report, einschließlich der Namen der Datendateien, befindet sich in der Reportdatei. Für die Anzeige des Reports reicht die Funktion **report4do** aus; die internen Funktionen erledigen den Rest.

Würden Sie dieselbe Anwendung für Windows schreiben, sähe der Code folgendermaßen aus (Programm WREP1.C):

```
#include <windows.h>
#include "d4all.h"
#include "r4report.h"

#define IDM_DOREPORT 101

static char *reportName ;
long FAR PASCAL WndProc (HWND, UINT, WPARAM, LPARAM) ;

int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                    LPSTR lpszCmdParam, int nCmdShow)
{
     static char szAppName[] = "WREP1" ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;
     reportName = lpszCmdParam ;
```

```
     if (!hPrevInstance)
     {
          wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
          wndclass.lpfnWndProc   = WndProc ;
          wndclass.cbClsExtra    = 0 ;
          wndclass.cbWndExtra    = 0 ;
          wndclass.hInstance     = hInstance ;
          wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
          wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
          wndclass.hbrBackground = GetStockObject (WHITE_BRUSH) ;
          wndclass.lpszMenuName  = "MAINMENU" ;
          wndclass.lpszClassName = szAppName ;

          RegisterClass (&wndclass) ;
     }

     hwnd = CreateWindow (szAppName, "Anwendungsfenster",
           WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
           CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL)
;

     ShowWindow (hwnd, nCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

long FAR PASCAL WndProc (HWND hWnd, UINT message, WPARAM wParam
                                                    , LPARAM lParam)
{
     static CODE4 cb ;
     static REPORT4 *report ;

     switch (message)
     {
          case WM_COMMAND:
               switch( wParam )
               {
```

```
                    case IDM_DOREPORT:
                       report=report4retrieve(&cb, reportName, 1, NULL) ;
                       if( report )
                       {
                          report4parent( report, hWnd ) ;
                          report4toScreen( report, 1 ) ;
                          report4do( report ) ;
                       }
                       break ;
                }
                break ;
           case WM_CREATE:
                d4init( &cb ) ;
                break ;
           case CRM_REPORTCLOSED: /* Von report4do gemeldet, s. API */
                report4free( report, 1, 1 ) ;
                break ;
           case WM_DESTROY:
                d4init_undo( &cb ) ;
                PostQuitMessage (0) ;
                return 0 ;
      }

      return DefWindowProc (hWnd, message, wParam, lParam) ;
}
```

Windows-Resource-Datei für das Menü (WREP1.RC):

```
MAINMENU MENU
BEGIN
   MENUITEM "&Report anzeigen", 101
END
```

Windows-Definitionsdatei (WREP1.DEF):

```
DESCRIPTION    'WREP1 CodeReporter-Beispiel'
EXETYPE        WINDOWS
STUB           'WINSTUB.EXE'
CODE           PRELOAD MOVEABLE DISCARDABLE
DATA           PRELOAD FIXED MULTIPLE
HEAPSIZE       4096
STACKSIZE      15000
```

Generierten Code verwenden

Für jede beliebige Reportdatei kann CodeReporter Quellcode generieren. Der fest codierte Quellcode für diese Reports wird in Dateien abgelegt, die kompiliert und mit allen Anwendungen gelinkt werden können, die den Report einsetzen. Dazu sind von seiten der Anwendung lediglich ein Prototyp der Funktion, die die **REPORT4**-Struktur erzeugt, sowie ein entsprechender Funktionsaufruf erforderlich. Danach werden die „do"- und „free"-Abfolgen genauso ausgeführt, als wäre der Report mit der Funktion **report4retrieve** geladen worden.
Das Programm REP2.C verwendet von CodeReporter generierten Quellcode:

```
#include "d4all.h"

#ifdef __TURBOC__
   extern unsigned _stklen = 10000 ;
#endif

/* Prototyp der von CodeReporter generierten Funktion */
REPORT4 *buildReport( CODE4 *, int) ;

void main( void )
{
   CODE4  cb ;
   REPORT4 *report ;

   d4init( &cb ) ;

   report = buildReport( &cb, 1 ) ;

   if( report )
   {
      report4do( report ) ;
      report4free( report, 1, 1) ;
   }
   d4init_undo( &cb ) ;
   return ;
}
```

REP2.C benutzt eine Reportdatei, die als Quellcode gespeichert wurde. Wie oben erläutert, wird die von CodeReporter generierte Funktion **buildReport** zum Aufbau des Reports verwendet. Der Code für diese Funktion muß mit REP2.C gelinkt werden, um die ausführbare Datei zu erzeugen.

Der Prototyp der Funktion **buildReport** sollte vor dem Abschnitt *main* stehen. Er wird verwendet, um die Parameter und den Rückgabewert der von CodeReporter erzeugten Funktion ordnungsgemäß an den Compiler zu übergeben. Der Prototyp der von CodeReporter generierten Funktion hat folgendes Muster;

```
REPORT4 *buildReport(CODE4 *cb, int openFiles);
```

In derselben Weise läßt sich der generierte Quellcode auch in einer Windows-Anwendung einsetzen; dazu dient das Programm WREP2.C. Sie können den Report laden und ausgeben, indem Sie einfach die **report4retrieve**-Zeile gegen den Funktionsnamen in der generierten Quellcode-Reportdatei austauschen.
Ausschnitt aus der Quellcodedatei WREP2.C für Windows:

```
...
long FAR PASCAL WndProc (HWND hWnd, UINT message, WPARAM wParam
                                                    , LPARAM lParam)
{
    static CODE4 cb ;
    static REPORT4 *report ;

    switch (message)
    {
        case WM_COMMAND:
            switch( wParam )
            {
                case IDM_DOREPORT:
                    report = buildReport( &cb, 1 ) ;
                    if( report )
                    {
                        report4parent( report, hWnd ) ;
                        report4toScreen( report, 1 ) ;
                        report4do( report ) ;
                    }
                    break ;
            }
            break ;
...
```

Einen Report ganz von vorn aufbauen

Da CodeReporter für jeden beliebigen Report Quellcode erzeugen kann, ist es wenig empfehlenswert, Reports ganz von vorn aufzubauen. Muß ein Report an

bestimmte Anforderungen angepaßt werden, kann dazu als Ausgangspunkt der Quellcode eines bereits vorhandenen Reports verwendet werden.

Muß ein Report tatsächlich per Hand ganz neu aufgebaut werden, halten Sie sich bitte an die folgenden Schritte:

1. Stellen Sie zunächst die Relation her. Dabei können Sie entweder die Funktionen des Relationsmoduls von CodeBase 5 einsetzen oder eine Relation mit **relate4retrieve** aus einer Relationsdatei laden.
2. Initialisieren Sie die Reportstruktur durch einen Aufruf an **report4init**. Hierdurch wird intern Speicher reserviert und zahlreiche Voreinstellungen vorgenommen.
3. Stellen Sie mit **report4querySet** und **report4sortSet** die entsprechenden Abfrage- und/oder Sortierbedingungen her.
4. Verändern Sie die Voreinstellungen eventuell mit den folgenden Funktionen: **report4caption, report4currency, report4decimal, report4hardResets, report4margins, report4pageSize, report4separator** und **report4titlePage**.
5. Legen Sie die Gruppen des Reports mit **group4create** an. Da die Bereiche Seitenkopf/-fuß und Titel/Schluß automatisch von **report4init** angelegt werden, brauchen Sie sie nicht zu erzeugen.
6. Legen Sie die Gruppenbereiche mit **area4create** an.
7. Erzeugen Sie mit der CodeBase-5-Funktion **expr4calc_create** reportweite Berechnungen (falls nötig).
8. Erzeugen Sie die Ausgabeobjekte unter Verwendung der entsprechenden Funktionen, und stellen Sie sie in die gewünschten Bereiche.

Nachdem Sie den Report auf diese Weise aufgebaut haben, können Sie zu seiner Ausgabe die übliche „load“-, „do“- und „free“-Abfolge verwenden.

Angepaßte Ausgabetreiber

Das CodeReporter-API enthält zwei Treiber für die Plattformen Microsoft Windows und MS-DOS. Sollen die Reportfunktionen auf anderen Plattformen verwendet oder eine angepaßte Anzeigebibliothek (z. B. CodeScreens) eingesetzt werden, muß der Entwickler eine eigene, **report4do** entsprechende, Funktion erstellen.

Die Hauptfunktionen eines angepaßten Ausgabetreibers sind:

- **report4pageInit**
- **report4generatePage**
- **report4pageObjFirst**
- **report4pageObjNext**
- **report4pageFree**

Um den ordnungsgemäßen Einsatz dieser Funktionen zu gewährleisten, müssen für die Ausgabe der Reportdaten die folgenden Strukturen verwendet werden:

OBJECT4

Beschreibung Diese Struktur beschreibt ein ausgewertetes Ausgabeobjekt. Die **report4pageObj**-Funktionen geben einen Zeiger auf diese Struktur zurück.

Elemente objtype Dieses Flag zeigt den Typ des aktuellen Ausgabeobjekts an. *objtype* kann als Wert jede der folgenden Konstanten annehmen:

obj4type_field	*info* speichert den Feldinhalt als Text.
obj4type_expr	*info* speichert den Inhalt eines Ausdrucks als Text.
obj4type_total	*info* speichert den Wert einer Summe als Text.
obj4type_text	*info* speichert statischen Text.
obj4type_hline	*info* ist leer. Die waagerechte Linie wird mit *x, y, w* und *h* beschrieben.
obj4type_vline	*info* ist leer. Die senkrechte Linie wird mit *x, y, w* und *h* beschrieben.
obj4type_frame	*info* enthält zwei Bytes, mit denen Füllung und runde Ecken bestimmt werden. Ist das linke Byte „1“, ist der Rahmen gefüllt. Ist das rechte Byte „1“, hat der Rahmen runde Ecken.

alignment	Ausgabeobjekte, die als Ergebnis Text haben (*obj4type_field, obj4type_expr, obj4type_text, obj4type_total*), werden in bezug auf das Objekt links-, rechtsbündig bzw. mittig ausgerichtet. Mögliche Werte für *alignment* sind: justify4right — Rechtsbündige Ausrichtung. justify4left — Linksbündige Ausrichtung. justify4center — Mittige Ausrichtung.
x	bezeichnet die waagerechte Position der linken oberen Ecke des Ausgabeobjekts in tausendstel Zoll.
y	bezeichnet die senkrechte Position bis zur linken oberen Ecke des Ausgabeobjekts in tausendstel Zoll, gemessen vom oberen Blattrand.
w	gibt die Breite des Ausgabeobjekts in tausendstel Zoll an.
h	gibt die Höhe des Ausgabeobjekts in tausendstel Zoll an.
info_len	ist die Größe der in *info* für das Ausgabeobjekt abgelegten Daten.
info	Die Daten, die für das Ausgabeobjekt ausgegeben werden sollen.
style_index	ist ein Index für das Seitenlayout des Reports.

Siehe auch **report4pageObjFirst, report4pageObjNext, style4index**

STYLE4

Beschreibung Diese Struktur enthält die für die Beschreibung der Schriftart, Farbe und Attribute der Reportstyles erforderlichen Informationen.

Mit **style4lookup** und **style4index** erhalten Sie Zeiger auf die entsprechenden Style-Strukturen.

Elemente	name	ist ein null-terminiertes Zeichenarray, das den Namen des Styles enthält, wie er in Code-Reporter vorhanden ist.
	lfont	ist ein Zeiger auf eine Windows-LOGFONT-Struktur, die die vom Style verwendete Schriftart beschreibt.
	color	Dabei handelt es sich um einen vorzeichenlosen **(long)**-Wert, der die für den Style eingestellten RGB-Farben speichert, ein Byte für den Rotwert, eines für den Grünwert und eines für den Blauwert. Jedes Byte kann einen Wert von 0 bis 255 haben und so die Farbschattierung anzeigen. Die einzelnen Einstellungen lassen sich über folgende Makros abrufen: R4GETRVALUE(rgb) R4GET-GVALUE(rgb) R4GETBVALUE(rgb).
	point_size	Dabei handelt es sich um die Punktgröße der im Style verwendeten Schriftart.
	codes_before_len	ist die Länge der Codezeilen, auf die *codes_before* zeigt.
	codes_after_len	ist die Länge der Codezeilen, auf die *codes_after* zeigt.
	codes_before	Dabei handelt es sich um ein Zeichenarray mit den Druckersteuersequenzen im Seitenlayout, die die Druckattribute aktivieren.
	codes_after	Dabei handelt es sich um ein Zeichenarray mit den Druckersteuersequenzen im Seitenlayout, die die Druckattribute deaktivieren.

Siehe auch **style4index, style4create**

Treiber-Shell einsetzen

Die Grundstruktur eines Reportausgabetreibers ist im wesentlichen für alle Umgebungen und Interface-Bibliotheken gleich. Es folgt der kommentierte Quellcode (DRSHELL.C) als Beispiel für eine angepaßte Interface-Treiber-Shell.

```
int S4FUNCTION report4doDriverShell( REPORT4 *report )
{
   int rc, error ;
   OBJECT4 *obj ;
   STYLE4  *style ;
   /* Seitenstruktur initialisieren */
   if( report4pageInit( report ) < 0 )
      return -1 ;

   error = 0 ;
   while( (rc=report4generatePage(report)) >= 0 ) /* Seite füllen */
   {
      if( rc == 2 ) /* letzte Seite wurde erreicht*/
         break ;
      for( obj = report4pageObjFirst( report ); obj != NULL && !error
                               ; obj = report4pageObjNext( report ) )
      {
         /* alle ausgewerteten Ausgabeobjekte des Reports
          * durchlaufen */
         switch( obj->objtype )
         {
            case obj4type_field:
            case obj4type_expr:
            case obj4type_total:
            case obj4type_text:
               /* Text-Ausgaberoutine */

               if( (style = style4index( report, obj->style_index ))
                                                            == NULL)
               {
                 obj = NULL ;
                 error = 1 ;
                 break ;
               }
               /* Informationen über den Style des Objekts ermitteln
                * mit style->color, style->lfont und style->point_
                * size entsprechenden Ausgabefont aufbauen */
```

```
            if( report->output_handle == 1 )
            {
               /* Bildschirmausgabe */
               /* Koordinaten obj->x, obj->y einnehmen
                * und mit Bildschirm-Interfacefunktion den Text
                * ausgeben, auf den obj->info mit einer Länge
                * von obj->info_len zeigt, mit obj->w und obj->h,
                * falls erforderlich, Zeilenumbruch und
                * obj->alignment steuern */
            }
            else
            {
               /* Ausgabe auf einen Drucker */

               /* Koordinaten obj->x, obj->y einnehmen
                * und mit Drucker-Interfacefunktion den Text
                * ausgeben, auf den obj->info mit einer Länge
                * von obj->info_len zeigt, mit obj->w und obj->h,
                * falls erforderlich, Zeilenumbruch und
                * obj->alignment steuern */
            }
            break ;
         case obj4type_hline:
         case obj4type_vline:
            /* Routine zum Linienzeichnen */
            if( (style = style4index( report, obj->style_index ))
                                                          == NULL)
            {
              obj = NULL ;
              error = 1 ;
              break ;
            }
            /* mit style->color Linienfarbe einstellen */

            if( report->output_handle == 1 )
            {
               /* Koordinaten obj->x, obj->y einnehmen und einen
                * Rahmen mit Füllung zu obj->x+obj->w, obj->y+
                * obj->h zeichnen */
            }
            else
            {
```

```
            /* Koordinaten obj->x, obj->y einnehmen und einen
             * Rahmen mit Füllung zu obj->x+obj->w, obj->y+
             * obj->h drucken */
         }
         break ;
      case obj4type_frame:
          /* Routine zum Rahmenzeichnen */
         if( (style = style4index( report, obj->style_index ))
                                                      == NULL)
         {
           obj = NULL ;
           error = 1 ;
           break ;
         }
         /* mit style->color Rahmenfarbe einstellen */

         if( *((char *)obj->info) == 1 )
         {
            /*  runde Ecken einstellen */
         }
         else
         {
            /* gewinkelte Ecken einstellen */
         }
         if( *( ((char *)obj->info)+1 ) == 1 )
         {
            /* Rahmen mit Füllung einstellen */
         }
         else
         {
            /* Rahmen ohne Füllung einstellen */
         }

         if( report->output_handle == 1 )
         {
            /* das entsprechende Rechteck
             * von obj->x, obj->y bis
             * obj->x+obj->w, obj->y+obj->h zeichnen */
         }
         else
         {
            /* das entsprechende Rechteck
```

```
                    * von obj->x, obj->y bis
                    * obj->x+obj->w, obj->y+obj->h drucken */
                }
               break ;
            default:
                /* alle anderen Objekttypen ignorieren */
         }
      }
      /* Ausgabegerät für neue Seite freigeben */
   }
   report4pageFree( report ) ;
   if( report->code_base->error_code < 0 || error)
      return -1 ;
   return 0 ;
}
```

Wie aus den Codezeilen ersichtlich, durchläuft der Treiber alle ausgewerteten Ausgabeobjekte und gibt sie einzeln aus, und zwar für jede Seite des Reports.

Der Code geht davon aus, daß von jeder Stelle der Seite aus in das Ausgabegerät geschrieben werden kann. Erfolgt die Ausgabe an das Gerät zeilenorientiert (wie bei einem Nadeldrucker), müssen die Ausgabeobjekte möglicherweise zwischengespeichert werden.

Dazu können Sie im Speicher einen Puffer von der Größe des Ausgabegeräts einrichten und die Ausgabeobjekte darin nach und nach ablegen. Zeigt **report4pageObjNext** an, daß alle Objekte der Seite ausgegeben worden sind, kann die gepufferte Kopie dieser Seite ausgegeben werden.

Eine weitere Möglichkeit besteht darin, die Objekte auf der Seite zu durchlaufen und eine verkettete Liste der Objekte zu erstellen, die nach senkrechter und waagerechter Position sortiert wird. Enthält die Seite keine weiteren Objekte, können die Elemente anhand dieser Sortierliste dann zeilenweise ausgegeben werden.

Diese wenigen Beispiele lassen erkennen, daß das Erstellen eines maßgeschneiderten Reports mit dem CodeReporter-API ein äußerst einfaches wie auch höchst kompliziertes Unterfangen darstellen kann. In den meisten Fällen lassen sich aus einer Kombination von variabel codierten Reports und/oder von CodeReporter generiertem Quellcode komplexe Reports erstellen, ohne daß Sie großartig von Hand codieren müssen.

area4-Funktionen

Die **area4**-Funktionen dienen zur Erstellung von Reportausgabebereichen, in die Ausgabeobjekte gesetzt werden können. Darüber hinaus werden diese Funktionen verwendet, um die in einem Reportbereich vorhandenen Ausgabeobjekte zu durchlaufen.

area4create

Aufruf AREA4 *area4create(GROUP4 *group, long height, short isHeader, char *suppressExpr)

Beschreibung Diese Funktion erzeugt für die angegebene Gruppe einen Reportkopf- bzw. -fußbereich, in den Ausgabeobjekte plaziert werden können.

Parameter

group	Ein Zeiger auf die Gruppe, mit der der neue Reportbereich verbunden ist.
height	Dieser **(long)**-Wert gibt die Höhe des neuen Reportbereichs in tausendstel Zoll an.
isHeader	Dieses logische Flag bestimmt, ob der neue Reportbereich mit dem Kopf bzw. dem Fuß der Gruppe verbunden werden soll. Ist *isHeader* nicht-Null (wahr), wird der neue Bereich mit dem Gruppenkopf verbunden. Ist *isHeader* Null (falsch), wird der neue Bereich mit dem Gruppenfuß verbunden.
suppressExpr	Hierbei handelt es sich um ein null-terminiertes Zeichenarray, das auf einen logischen dBASE-Ausdruck zeigt, welcher bestimmt, ob der Reportbereich unterdrückt wird oder nicht, wenn eine Gruppenrücksetzbedingung eintritt. Bei einer

Gruppenrücksetzbedingung wird *suppressExpr* ausgewertet. Ist der Wert wahr, wird der Reportbereich für die Gruppenrücksetzbedingung nicht ausgegeben, ist der Wert falsch, erfolgt die Ausgabe. Wenn *suppressExpr* Null ist oder auf ein Array mit Leerzeichen zeigt, wird der neue Bereich niemals unterdrückt.

area4create legt eine Kopie des Parameters *suppressExpr* an. Infolgedessen kann man den Speicher für *suppressExpr* nach dem Funktionsaufruf freigeben.

Rückgaben Falls erfolgreich, gibt **area4create** einen **AREA4**-Zeiger zurück. Konnte der Bereich nicht erstellt werden, gibt **area4create** einen Null-Wert zurück.

Siehe auch **area4free, group4create, group4titleSummary**

area4free

Aufruf void area4free(AREA4 *area)

Beschreibung Diese Funktion entfernt einen angegebenen Bereich aus dem Report. Darüber hinaus wird der mit dem Reportbereich verbundene Speicher freigegeben; alle Ausgabeobjekte in dem Reportbereich werden gelöscht und freigegeben.

Parameter area Dieser **AREA4**-Zeiger bestimmt den zu löschenden Reportbereich.

Siehe auch **area4create, group4free**

area4numObjects

Aufruf int area4numObjects(AREA4 *area)

Beschreibung Diese Funktion gibt die aktuelle Anzahl der im angegebenen Bereich vorhandenen Ausgabeobjekte zurück.

Parameter area Dieser **AREA4**-Zeiger identifiziert den Bereich.

Rückgaben 0 Der angegebene Bereich enthält keine Ausgabeobjekte.

Nicht-Null Die Anzahl der im angegebenen Bereich vorhandenen Ausgabeobjekte.

Siehe auch **area4objFirst, area4objNext, area4objLast, area4objPrev**

area4objFirst

Aufruf OBJ4 *area4objFirst(AREA4 *area)

Beschreibung Diese Funktion ruft einen Zeiger auf das erste Ausgabeobjekt im angegebenen Bereich ab. In Verbindung mit **area4objNext** durchläuft sie die Ausgabeobjekte in einem Bereich.

Parameter area Ein Zeiger auf den Bereich, der die Ausgabeobjekte enthält.

Rückgaben Nicht-Null Ein **OBJ4**-Zeiger auf das erste Ausgabeobjekt in dem Reportbereich wird zurückgegeben.

0 Fehler. *area* ist ungültig, oder der angegebene Bereich enthält keine Ausgabeobjekte.

Siehe auch **area4objNext, area4objLast**

area4objLast

Aufruf OBJ4 *area4objLast(AREA4 *area)

Beschreibung Diese Funktion ruft einen Zeiger auf das letzte Ausgabeobjekt im angegebenen Bereich ab. In Verbindung mit **area4objPrev** durchläuft sie die Ausgabeobjekte in einem Bereich rückwärts.

Parameter

area	Ein Zeiger auf den Bereich, der die Ausgabeobjekte enthält.

Rückgaben

Nicht-Null	Ein **OBJ4**-Zeiger auf das letzte Ausgabeobjekt in dem Reportbereich wird zurückgegeben.
0	Fehler. *area* ist ungültig, oder der angegebene Bereich enthält keine Ausgabeobjekte.

Siehe auch **area4objPrev, area4objFirst**

area4objNext

Aufruf OBJ4 *area4objNext(AREA4 *area)

Beschreibung Diese Funktion ruft einen Zeiger auf das nächste Ausgabeobjekt im angegebenen Bereich ab. In Verbindung mit **area4objFirst** durchläuft sie die Ausgabeobjekte in einem Bereich.

Parameter

area	Ein Zeiger auf den Bereich, der die Ausgabeobjekte enthält.

Rückgaben

Nicht-Null	Ein **OBJ4**-Zeiger auf das nächste Ausgabeobjekt in dem Reportbereich wird zurückgegeben.
0	Fehler. *area* ist ungültig, oder das zuletzt abgerufene Ausgabeobjekt war das letzte in dem Bereich.

Siehe auch	**area4objFirst, area4objPrev**

area4objPrev

Aufruf	OBJ4 *area4objPrev(AREA4 *area)	
Beschreibung	Diese Funktion ruft einen Zeiger auf das vorhergehende Ausgabeobjekt im angegebenen Bereich ab. In Verbindung mit **area4objLast** durchläuft sie die Ausgabeobjekte einem Bereich rückwärts.	
Parameter	area	Ein Zeiger auf den Bereich, der die Ausgabeobjekte enthält.
Rückgaben	Nicht-Null	Ein **OBJ4**-Zeiger auf das vorhergehende Ausgabeobjekt in dem Reportbereich wird zurückgegeben.
	0	Fehler. *area* ist ungültig, oder das zuletzt abgerufene Ausgabeobjekt war das erste in dem Bereich.
Siehe auch	**area4objLast, area4objNext**	

area4pageBreak

Aufruf	void area4pageBreak(AREA4 *area, int allowBreaks)	
Beschreibung	Mit dieser Funktion werden in einem angegebenen Bereich Seitenumbrüche zugelassen oder verhindert. Wird diese Funktion nicht aufgerufen, sind Seitenumbrüche innerhalb der Bereiche nicht möglich.	
Parameter	area	Mit diesem **AREA4**-Zeiger wird der Bereich bestimmt.

	allowBreaks	Ist *allowBreaks* Null (falsch), wird innerhalb des Bereichs kein Seitenumbruch zugelassen. Bei einem Seitenumbruch innerhalb der Gruppe wird diese auf der folgenden Seite ausgegeben. Ist *allowBreaks* nicht-Null (wahr), kann ein Seitenumbruch innerhalb des Reportbereichs vorkommen. Dabei wird der Bereich, soweit möglich, auf der aktuellen Seite ausgegeben, während der Rest auf der folgenden Seite erscheint.
Rückgaben	>= 0	Gibt die vorher gültige Einstellung zum Seitenumbruch zurück.
	< 0	*area* oder *allowBreaks* sind ungültig.

group4-Funktionen

Mit den **group4**-Funktionen definiert man die Operationen, die innerhalb einer Untermenge der zusammengesetzten Datendatei ausgeführt werden. Diese Operationen haben hauptsächlich mit der Ausgabe von Reportbereichen zu tun. Mit diesen Funktionen werden darüber hinaus besondere Merkmale der Ausgabe definiert, das Austauschen von Kopf-/Fußbereichen gegen die der Seite, das Zurücksetzen von Seite und Seitenzahl usw.

Die **group4**-Funktionen werden ebenfalls zum Durchlaufen der mit der Gruppe verbundenen Kopf- und Fußbereiche verwendet.

group4create

Aufruf GROUP4 *group4create(REPORT4 *report, char *name, char *resetExpr)

Beschreibung **group4create** erzeugt im angegebenen Report eine neue Gruppe. Per Voreinstellung handelt es sich bei der neuen Gruppe um die am weitesten außen gelegene.

Parameter

report — Ein Zeiger auf den Report, zu dem die neue Gruppe hinzugefügt wird.

name — Ein null-terminiertes Zeichenarray, das eine nur einmal vorkommende beschreibende Bezeichnung für die Gruppe enthält. Ist dieser Parameter Null, wird der Name der Gruppe auf „Gruppe *n*“ voreingestellt, wobei *n* die Position der erzeugten Gruppe angibt. **group4create** legt eine Kopie von *name* an.

resetExpr — Ein null-terminiertes Zeichenarray mit einem dBASE-Ausdruck, der bestimmt, wann die Gruppe zurückgesetzt wird. Das Reportmodul wertet

		diesen Ausdruck für jeden Datensatz der zusammengesetzten Datendatei aus; tritt eine Wertänderung ein, wird die Gruppe zurückgesetzt und der Bereich für die Gruppe ausgegeben. Ist *resetExpr* Null, wird die Gruppe für jeden Datensatz in der zusammengesetzten Datendatei zurückgesetzt.
Rückgaben	0	Beim Anlegen der Gruppe hat sich ein Fehler ereignet.
	Nicht-Null	Es wird ein Zeiger auf die erfolgreich angelegte Gruppe zurückgegeben.

Siehe auch **area4create, group4free**

group4footerFirst

Aufruf AREA4 *group4footerFirst(GROUP4 *group)

Beschreibung Diese Funktion gibt einen **AREA4**-Zeiger auf den ersten für die Gruppe angelegten Fußbereich zurück.

Parameter group — Ein **GROUP4**-Zeiger für die Gruppe.

Rückgaben **group4footerFirst** gibt einen **AREA4**-Zeiger auf den ersten für die Gruppe angelegten Fußbereich zurück. Wenn die Gruppe keinen Fußbereich hat, gibt diese Funktion Null zurück.

Siehe auch **group4headerFirst, group4footerNext, group4footerPrev**

group4footerNext

Aufruf AREA4 *group4footerNext(GROUP4 *group, AREA4 *area)

Beschreibung Diese Funktion gibt einen Zeiger auf den in der angegebenen Gruppe erzeugten Fußbereich hinter dem angegebenen Fußbereich zurück. Sie wird im allgemeinen in Verbindung mit **group4footerFirst** verwendet, um die Fußbereiche einer Gruppe zu durchlaufen.

Parameter

group	Ein Zeiger auf die Gruppe mit dem Bereich *area*.
area	Ein Zeiger auf den Fußbereich, von dem aus der als nächster angelegte Fußbereich lokalisiert wird.

Rückgaben Diese Funktion gibt einen **AREA4**-Zeiger auf den Fußbereich zurück, der nach dem Fußbereich *area* angelegt wurde. Ist *area* der letzte für die Gruppe angelegte Fußbereich oder ist *area* Null, gibt **group4footerNext** Null zurück.

Achtung! Es kann zu unerwarteten Ergebnissen kommen, wenn der Parameter *area* kein gültiger Fußbereich für die Gruppe *group* ist.

Siehe auch **group4footerFirst, group4numFooters**

group4footerPrev

Aufruf AREA4 *group4footerPrev(GROUP4 *group, AREA4 *area)

Beschreibung Diese Funktion gibt einen Zeiger auf den in der angegebenen Gruppe unmittelbar vor dem angegebenen Bereich erzeugten Fußbereich zurück. Sie wird im allgemeinen in Verbindung mit **area4lastFooter** verwendet, um die Fußbereiche einer Gruppe zu durchlaufen.

Parameter

group	Ein Zeiger auf die Gruppe mit dem Fußbereich *area*.
area	Ein Zeiger auf den Fußbereich, von dem aus der unmittelbar vorher angelegte Fußbereich lokalisiert wird.

Rückgaben Diese Funktion gibt einen **AREA4**-Zeiger auf den Fußbereich zurück, der vor dem Fußbereich *area* angelegt wurde. Ist *area* der letzte für die Gruppe angelegte Fußbereich oder ist *area* Null, gibt **group4footerPrev** Null zurück.

Achtung! Es kann zu unerwarteten Ergebnissen kommen, wenn der Parameter *area* kein gültiger Fußbereich für die Gruppe *group* ist.

Siehe auch **area4firstFooter, area4numFooters**

group4free

Aufruf void group4free(GROUP4 *group)

Beschreibung Diese Funktion löscht die angegebene Gruppe aus dem Report und gibt den damit verbundenen Speicher frei. Hat die Gruppe Kopf- und/oder Fußbereiche, werden diese mit **area4free**-Aufrufen entfernt.

Parameter group Der Zeiger auf die aus dem Report zu löschende Gruppe.

Siehe auch **group4create, area4free**

group4headerFirst

Aufruf AREA4 *group4headerFirst(GROUP4 *group)

Beschreibung Diese Funktion gibt einen **AREA4**-Zeiger auf den ersten für die Gruppe angelegten Kopfbereich zurück.

Parameter group Ein **GROUP4**-Zeiger auf die Gruppe.

Rückgaben **group4headerFirst** gibt einen **AREA4**-Zeiger auf den ersten für die Gruppe angelegten Kopfbereich zurück. Wenn die Gruppe keinen Kopfbereich hat, gibt diese Funktion Null zurück.

Siehe auch **group4footerFirst, group4headerNext, group4headerPrev**

group4headerNext

Aufruf AREA4 *group4headerNext(GROUP4 *group, AREA4 *area)

Beschreibung Diese Funktion gibt einen Zeiger auf den in der angegebenen Gruppe erzeugten Kopfbereich hinter dem angegebenen Kopfbereich zurück. Sie wird im allgemeinen in Verbindung mit **group4headerFirst** verwendet, um die Kopfbereiche einer Gruppe zu durchlaufen.

Parameter

group	Ein Zeiger auf die Gruppe mit dem Bereich *area*.
area	Ein Zeiger auf den Kopfbereich, von dem aus der als nächster angelegte Kopfbereich lokalisiert wird.

Rückgaben Diese Funktion gibt einen **AREA4**-Zeiger auf den Kopfbereich zurück, der nach dem Kopfbereich *area* angelegt wurde. Ist *area* der letzte für die Gruppe angelegte Kopfbereich oder ist *area* Null, gibt **group4headerNext** Null zurück.

Achtung! Es kann zu unerwarteten Ergebnissen kommen, wenn der Parameter *area* kein gültiger Kopfbereich für die Gruppe *group* ist.

Siehe auch **group4headerFirst, group4numHeaders**

group4headerPrev

Aufruf AREA4 *group4headerPrev(GROUP4 *group, AREA4 *area)

Beschreibung Diese Funktion gibt einen Zeiger auf den in der angegebenen Gruppe unmittelbar vor dem angegebenen Bereich erzeugten Kopfbereich zurück. Sie wird im allgemeinen in Verbindung mit **area4lastHeader** verwendet, um die Kopfbereiche einer Gruppe zu durchlaufen.

Parameter

group	Ein Zeiger auf die Gruppe mit dem Kopfbereich *area*.
area	Ein Zeiger auf den Kopfbereich, von dem aus der unmittelbar vorher angelegte Kopfbereich lokalisiert wird.

Rückgaben Diese Funktion gibt einen **AREA4**-Zeiger auf den Kopfbereich zurück, der vor dem Kopfbereich *area* angelegt wurde. Ist *area* der letzte für die Gruppe angelegte Kopfbereich oder ist *area* Null, gibt **group4headerPrev** Null zurück.

Achtung! Es kann zu unerwarteten Ergebnissen kommen, wenn der Parameter *area* kein gültiger Kopfbereich für die Gruppe *group* ist.

Siehe auch **area4firstHeader, area4numHeaders**

group4numFooters

Aufruf int group4numFooters(GROUP4 *group)

Beschreibung Diese Funktion gibt die aktuelle Anzahl der Fußbereiche für die angegebene Gruppe zurück.

Parameter

group	Dieser **GROUP4**-Zeiger identifiziert die Gruppe.

Rückgaben	0	Die angegebene Gruppe enthält keine Fußbereiche.
	Nicht-Null	Die Anzahl der für die angegebene Gruppe angelegten Fußbereiche.

Siehe auch **area4create, area4free**

group4numHeaders

Aufruf int group4numHeaders(GROUP4 *group)

Beschreibung Diese Funktion gibt die aktuelle Anzahl der Kopfbereiche für die angegebene Gruppe zurück.

Parameter	group	Dieser **GROUP4**-Zeiger identifiziert die Gruppe.

Rückgaben	0	Die angegebene Gruppe enthält keine Kopfbereiche.
	Nicht-Null	Die Anzahl der für die angegebene Gruppe angelegten Kopfbereiche.

Siehe auch **area4create, area4free**

group4repeatHeader

Aufruf int group4repeatHeader(GROUP4 *group, int repeatHeader)

Beschreibung Mit Hilfe dieser Funktion wird bestimmt, ob der/die Kopfbereich(e) der angegebenen Gruppe oben auf einer neuen Seite ausgegeben werden soll(en), falls innere Gruppen sich über einen Seitenumbruch hinweg erstrecken.

Die Funktion wirkt sich nicht auf den Seitenkopf aus. Wird sie mit *repeatHeader* gleich „1“ aufgerufen, wird der Kopf unterhalb des Seitenkopfes (und aller höherrangigen wiederholten

Kopfbereiche) und vor dem ersten Kopfbereich einer untergeordneten Gruppe ausgegeben.

Parameter

group	Die Gruppe, für die die Option „Kopf wiederholen“ eingerichtet wird.
repeatHeader	Ist *repeatHeader* gleich „1“, wird der Kopf auf der folgenden Seite wiederholt. Enthält er den Wert „0“ (Voreinstellung), wird der Kopfbereich lediglich beim Eintreten einer Rücksetzbedingung ausgegeben.

Hinweis: Wird diese Funktion nicht aufgerufen, wird der Kopf nicht wiederholt.

Rückgaben

>= 0	Die vorhergehende Einstellung von *repeatHeader* wird zurückgegeben.
< 0	Fehler.

Siehe auch **group4swapHeader**

group4resetExprSet

Aufruf

```
int group4resetExprSet( GROUP4 *group, char *resetExpr )
```

Beschreibung Mit dieser Funktion läßt sich der Rücksetzausdruck der angegebenen Gruppe verändern.

Parameter

group	Die Gruppe, für die der Rücksetzausdruck eingerichtet wird.
resetExpr	Ein null-terminiertes Zeichenarray mit dem neuen dBASE-Ausdruck, der bestimmt, wann die Bereiche der Gruppe ausgegeben werden. Ist *resetExpr* Null, wird der Gruppenrücksetzausdruck gelöscht und die Gruppe für jeden zusammengesetzten Datensatz zurückgesetzt.

Rückgaben 0 Der Gruppenrücksetzausdruck wurde erfolgreich eingerichtet.

< 0 Fehler.

Siehe auch **group4create**

group4resetPage

Aufruf int group4resetPage(GROUP4 *group, int resetPage)

Beschreibung Mit Hilfe dieser Funktion wird bestimmt, ob vor der Ausgabe des Kopfbereichs der angegebenen Gruppe ein Seitenumbruch erzwungen werden soll.

Parameter

group Die Gruppe, für die die Option „Seite zurücksetzen“ eingerichtet wird.

resetPage Dieser Parameter kann zwei Einstellungen haben:

1 Ist *resetPage* auf „1“ eingestellt, erfolgt jedesmal ein erzwungener Seitenumbruch, wenn die Gruppe auf eine Rücksetzbedingung trifft.

0 Ist *resetPage* auf „0“ eingestellt (Voreinstellung), werden keine besonderen Maßnahmen ergriffen, um den Gruppenkopf auf der folgenden Seite auszugeben.

Hinweis: Wird diese Funktion nicht aufgerufen, wird die Seite bei der Zurücksetzung der Gruppe nicht zurückgesetzt.

Rückgaben >= 0 Die vorhergehende Einstellung von *resetPage* wird zurückgegeben.

< 0 Fehler.

Siehe auch **group4resetPageNum**

group4resetPageNum

Aufruf int group4resetPageNum(GROUP4 *group, int resetPageNum)

Beschreibung Mit Hilfe dieser Funktion wird bestimmt, ob vor der Ausgabe des Kopfbereichs der angegebenen Gruppe ein Seitenumbruch erzwungen werden soll. Ist die Option „Seitenumbruch“ aktiv, wird die Seitenzahl auf „1“ zurückgesetzt.

Parameter

group — Die Gruppe, für die die Option „Seitenzahl zurücksetzen“ eingerichtet wird.

resetPageNum — Dieser Parameter kann zwei Einstellungen haben:

1 Ist *resetPageNum* auf „1“ eingestellt, erfolgt jedesmal ein erzwungener Seitenumbruch, wenn die angegebene Gruppe auf eine Rücksetzbedingung trifft. Des weiteren wird die Seitenzahl auf „1“ gesetzt.

0 Ist *resetPageNum* auf „0“ eingestellt (Voreinstellung), werden keine besonderen Maßnahmen ergriffen, den Gruppenkopf auf der folgenden Seite auszugeben; die Seitenzahl wird weiter hochgezählt.

Rückgaben

>= 0 — Die vorhergehende Einstellung von *resetPageNum* wird zurückgegeben.

< 0 — Fehler.

Siehe auch dBASE-Ausdruck PAGENO(), **group4resetPage**

group4swapFooter

Aufruf int group4swapFooter(GROUP4 *group, int swap)

Beschreibung Diese Funktion tauscht beim Zurücksetzen der Gruppe den Seitenfuß-Bereich gegen den Gruppenfuß-Bereich aus.

Tritt für die Gruppe eine Rücksetzbedingung ein, wird der Rest der aktuellen Seite übersprungen, und der/die Fußbereich(e) der angegebenen Gruppe wird/werden anstelle des üblichen Seitenfußes ausgegeben; der Seitenfuß wird nicht ausgegeben.

Wenn die Gruppe nicht zurückgesetzt wird und es ereignet sich ein Seitenumbruch, wird wie üblich der Seitenfuß ausgegeben.

Parameter

group Dieser Zeiger bezeichnet die Gruppe, deren Fußbereich(e) anstelle des Seitenfuß-Bereichs verwendet werden soll(en).

swap Dieses logische Flag bestimmt, ob die Gruppe ihren Fußbereich austauschen soll oder nicht. Mögliche Werte für *swap* sind:

1 Der Gruppenfuß wird ausgetauscht.

0 Der Gruppenfuß wird nicht ausgetauscht. Wird die Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben

>= 0 Der vorhergehende Wert von *swap* wird zurückgegeben.

< 0 Fehler.

Siehe auch **group4swapHeader, report4pageHeaderFooter**

group4swapHeader

Aufruf int group4swapHeader(GROUP4 *group, int swap)

Beschreibung Diese Funktion tauscht beim Zurücksetzen der Gruppe den Seitenkopf-Bereich gegen den Gruppenkopf-Bereich aus.

Tritt für die Gruppe eine Rücksetzbedingung ein, wird der Rest der aktuellen Seite übersprungen (die Fußbereiche aller inneren Gruppen werden noch ausgegeben), und der/die Kopfbereich(e) der angegebenen Gruppe wird/werden anstelle des üblichen Seitenkopfes oben auf der folgenden Seite ausgegeben; der Seitenkopf wird nicht ausgegeben.

Wenn die Gruppe nicht zurückgesetzt wird und es ereignet sich ein Seitenumbruch, wird wie üblich der Seitenkopf ausgegeben.

Parameter

group — Dieser Zeiger bezeichnet die Gruppe, deren Kopfbereich(e) anstelle des Seitenkopf-Bereichs verwendet werden soll(en).

swap — Dieses logische Flag bestimmt, ob die Gruppe ihren Kopfbereich austauschen soll oder nicht. Mögliche Werte für *swap* sind:

- 1 Der Gruppenkopf wird ausgetauscht.
- 0 Der Gruppenkopf wird nicht ausgetauscht. Wird die Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben

>= 0 — Der vorhergehende Wert von *swap* wird zurückgegeben.

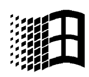
< 0 — Fehler.

Siehe auch **group4swapFooter, report4pageHeaderFooter**

obj4-Funktionen

Die **obj4**-Funktionen sind dazu da, innerhalb eines Reportbereichs Ausgabeobjekte zu erzeugen und zu modifizieren. Die Objekte werden mit der für ihren Typ vorgesehenen Funktion erzeugt und mit der entsprechenden Löschfunktion oder **obj4delete** aus dem Report entfernt.

Darüber hinaus stehen nach der Erstellung des Objekts besondere Formatier- und Layout-Funktionen zur Verfügung, mit denen sich das Erscheinungsbild des Ausgabeobjekts verändern läßt.



obj4bitmapStaticCreate

Aufruf OBJ4 *obj4bitmapStaticCreate(AREA4 *area, HANDLE hDIB,
long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt unter Verwendung eines Handles auf eine geräteunabhängige Bitmap unter Windows ein Grafikobjekt. Nachdem **obj4bitmapStaticCreate** ein Grafikausgabeobjekt erzeugt hat, darf der Programmierer das Handle auf die geräteunabhängige Bitmap (*hDIB*) nicht freigeben. Die Bitmap wird von **obj4bitmapStaticFree** automatisch freigegeben.

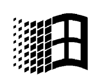
Um sie innerhalb der angegebenen Koordinaten unterzubringen, wird die Größe der Bitmap verändert; dabei bleiben ihre Proportionen nicht unbedingt erhalten.

Das Bild des statischen Grafikobjekts wird in der Reportdatei abgelegt.

Parameter area Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Grafikausgabeobjekt gesetzt werden soll.

	hDIB	Ein Handle auf eine geräteunabhängige Bitmap unter Windows.
	x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Grafikobjekts steht.
	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Grafikobjekts steht.
	width	Die horizontale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.
	height	Die vertikale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.
Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Grafikausgabeobjekt zurückgegeben.
	0	Fehler. Das Grafikausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4bitmapStaticFree, obj4bitmapFileCreate**



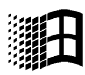
obj4bitmapStaticFree

Aufruf void obj4bitmapStaticFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein (mit **obj4bitmapStaticCreate** angelegtes) Grafikausgabeobjekt aus dem Report und gibt den mit der Bitmap und dem Ausgabeobjekt verbundenen Speicher frei.

Parameter obj — Zeiger auf das zu löschende statische Grafikausgabeobjekt.

Siehe auch **obj4delete, obj4bitmapStaticCreate, report4free**



obj4bitmapFileCreate

Aufruf OBJ4 *obj4bitmapFileCreate(AREA4 *area, char *fileName, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein statisches Grafikausgabeobjekt, indem sie eine vorhandene Windows-Bitmap-Datei (.BMP) öffnet und intern eine Kopie des Bildes anlegt. Nach Fertigstellung des Grafikausgabeobjekts wird die Bitmap-Datei geschlossen.

Um sie innerhalb der angegebenen Koordinaten unterzubringen, wird die Größe der Bitmap verändert; dabei bleiben ihre Proportionen nicht unbedingt erhalten.

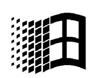
Wird ein Report mit einem solchen statischen Grafikobjekt gespeichert, wird lediglich ein Verweis auf den Dateinamen mit abgelegt. Das eigentliche Bild wird bei der Ausführung des Reports jeweils aus der Bitmap-Datei neu eingelesen.

Parameter

area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Grafikausgabeobjekt gesetzt werden soll.
fileName	Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Dateinamen einer Windows-Bitmap enthält. Sind Laufwerk und/oder Verzeichnis nicht vorhanden, geht das Programm vom aktuellen Laufwerk/Verzeichnis aus.
x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Grafikobjekts steht.
y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Grafikobjekts steht.
width	Die horizontale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.
height	Die vertikale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.

Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Grafikausgabeobjekt zurückgegeben.
	0	Fehler. Das Grafikausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4bitmapFileFree, obj4bitmapStaticCreate**



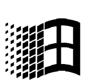
obj4bitmapFileFree

Aufruf void obj4bitmapFileFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein (mit **obj4bitmapFileCreate** angelegtes) Grafikausgabeobjekt aus dem Report und gibt den mit der Bitmap und dem Ausgabeobjekt verbundenen Speicher frei.

Parameter	obj	Zeiger auf das zu löschende statische Grafikobjekt, der von **obj4bitmapFileCreate** zurückgegeben wird.

Siehe auch **obj4bitmapFileCreate, obj4delete, report4free**



obj4bitmapFieldCreate

Aufruf OBJ4 *obj4bitmapFieldCreate(AREA4 *area, FIELD4 *field, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein dynamisches Grafikausgabeobjekt, indem sie die im Feld der Datendatei angegebene Windows-

Bitmap-Datei (.BMP) öffnet und intern eine Kopie des Bildes anlegt. Nach der Ausgabe des Grafikausgabeobjekts wird die Bitmap-Datei geschlossen.

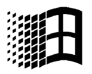
Um sie innerhalb der angegebenen Koordinaten unterzubringen, wird die Größe der Bitmap verändert; dabei bleiben ihre Proportionen nicht unbedingt erhalten.

Wird ein Report mit einem solchen dynamischen Grafikobjekt gespeichert, wird lediglich ein Verweis auf den Feldnamen mit abgelegt. Das eigentliche Bild wird bei jeder Ausgabe des Grafikausgabeobjekts aus der mit dem aktuellen Feldwert bezeichneten Bitmap-Datei neu eingelesen.

Parameter	area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Grafikausgabeobjekt gesetzt werden soll.
	field	Ein **FIELD4**-Zeiger auf das Feld einer Datendatei, das Laufwerk, Verzeichnis und Dateinamen einer Windows-Bitmap-Datei (.BMP) enthält. Werden Laufwerk und/oder Verzeichnis im Feld nicht genannt, geht das Programm vom aktuellen Laufwerk/Verzeichnis aus.
	x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Grafikobjekts steht.
	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Grafikobjekts steht.
	width	Die horizontale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.
	height	Die vertikale Ausdehnung des Grafikausgabeobjekts in tausendstel Zoll.
Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Grafikausgabeobjekt zurückgegeben.

	0	Fehler. Das Grafikausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4bitmapFieldFree, obj4delete**



obj4bitmapFieldFree

Aufruf void obj4bitmapFieldFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein (mit **obj4bitmapFieldCreate** angelegtes) dynamisches Grafikausgabeobjekt aus dem Report und gibt den mit der Bitmap und dem Ausgabeobjekt verbundenen Speicher frei.

Parameter	obj	Zeiger auf das zu löschende Grafikobjekt, der von **obj4bitmapFieldCreate** zurückgegeben wird.

Siehe auch **obj4bitmapFieldCreate, obj4delete, report4free**

obj4brackets

Aufruf int obj4brackets(OBJ4 *obj, int useBrackets)

Beschreibung Diese Funktion gibt an, ob das bezeichnete numerische Ausgabeobjekt bei negativen Zahlen Klammern verwenden soll oder nicht.

Parameter	obj	Ein Zeiger auf das numerische Objekt, das mit Klammern versehen werden soll.
	useBrackets	Dieser Parameter legt fest, ob negative Zahlen in Klammern eingeschlossen werden sollen.

useBrakkets kann einen der folgenden Werte annehmen:

1 Negative Zahlen werden in Klammern eingeschlossen (also (123)).

0 Negativen Zahlen wird das Minuszeichen vorangestellt (also -123).

Rückgaben >= 0 Die vorhergehende Einstellung von *useBrackets* wird zugegeben.

< 0 Fehler. *obj* oder *useBrackets* sind ungültig.

Siehe auch **obj4numericType, obj4displayZero**

obj4dataFieldSet

Aufruf

```
int obj4dataFieldSet( OBJ4 *obj, char *destField, char type,
                                    int length, int decimals )
```

Beschreibung Diese Funktion wird bei der Ausgabe eines Reports in eine Datendatei verwendet, um ein Objekt des Reports einem Feld der Zieldatendatei zuzuordnen. Diese Einstellung wird lediglich verwendet, wenn der Report mit **report4output** oder **report4toScreen** in eine Datendatei ausgegeben wird.

Parameter

obj Ein Zeiger auf das Ausgabeobjekt, das in eine Datendatei umgeleitet werden soll.

destField Ein null-terminiertes Zeichenarray, das den Namen des Datendateifeldes enthält, in das der Inhalt des Objekts gesetzt werden soll.

type Ein Großbuchstabe, der den Typ des Feldes beschreibt, in das der Inhalt des Ausgabeobjekts gesetzt werden soll. *type* kann folgende Werte annehmen: „C“ (Zeichen), „D“ (Datum), „L“ (Logisch), „N“ (Numerisch).

	length	Höchstanzahl der Zeichen des Inhalts des Ausgabeobjekts, die in die Ausgabedatendatei kopiert werden soll.
	decimals	Anzahl der Zeichen in *length*, die als Dezimalstellen reserviert werden soll. *decimals* wird lediglich verwendet, wenn *type* „N“ ist; ansonsten wird der Parameter ignoriert.
Rückgaben	0	Ausgabeobjekt erfolgreich mit dem Zielfeld verbunden.
	< 0	Fehler. *obj* oder *destField* sind ungültig.

Siehe auch **report4dataFileSet, report4dataGroup, d4create** von CodeBase 5.

obj4calcCreate

Aufruf OBJ4 *obj4calcCreate(AREA4 *area, EXPR4CALC *calc, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein Berechnungsausgabeobjekt; die verwendete Berechnung wurde mit der CodeBase-5-Funktion **expr4calc_create** erstellt.

Parameter	area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Berechnungsausgabeobjekt gesetzt werden soll.
	calc	Ein Zeiger auf die mit **expr4calc_create** erstellte Berechnung.
	x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Berechnungsobjekts steht.

	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Berechnungsobjekts steht.
	width	Die horizontale Ausdehnung des Berechnungsausgabeobjekts in tausendstel Zoll.
	height	Die vertikale Ausdehnung des Berechnungsausgabeobjekts in tausendstel Zoll.
Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Berechnungsausgabeobjekt zurückgegeben.
	0	Fehler. Das Berechnungsausgabeobjekt konnte nicht angelegt werden.

Siehe auch **expr4calc_create, obj4calcFree, obj4delete**

obj4calcFree

Aufruf void obj4calcFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht das Berechnungsausgabeobjekt aus dem Report und gibt den mit dem Ausgabeobjekt verbundenen Speicher frei.

Hinweis: **obj4calcFree** gibt weder den mit der eigentlichen Berechnung verbundenen Speicher frei noch wird die Berechnung gelöscht. Lassen Sie die Berechnung mit der CodeBase-5-Funktion **expr4calc_reset** oder mit **report4free** löschen.

Parameter obj — Zeiger auf das zu löschende Berechnungsobjekt.

Siehe auch **obj4calcCreate, obj4delete, report4free**, die CodeBase-5-Funktion **expr4calc_reset**

obj4dateFormat

Aufruf int obj4dateFormat(OBJ4 *obj, char *dateFormat)

Beschreibung Diese Funktion stellt das Format (auch als Schablone oder Maske bekannt) ein, das das Datumsausgabeobjekt bei der Ausgabe benutzt.

Hat das Ausgabeobjekt als Ergebnis keinen Datumswert, zeigt diese Funktion keine Wirkung.

Wenn die Funktion nicht aufgerufen wird und das Ausgabeobjekt als Ergebnis einen Datumswert aufweist, wird das für den Report voreingestellte Datumsformat verwendet (siehe **report4-dateFormat**).

Parameter

obj	Ein Zeiger auf das Ausgabeobjekt, für das das Datumsformat eingerichtet wird.
dateFormat	Ein null-terminiertes Zeichenarray, das das für das Ausgabeobjekt geltende neue Datumsformat enthält. Ist *dateFormat* Null, wird das aktuelle Datumsformat ignoriert und das voreingestellte Format des Reports verwendet. **obj4dateFormat** legt eine Kopie von *dateFormat* an.

Rückgaben

0	Das Datumsformat für das Ausgabeobjekt wurde erfolgreich eingerichtet.
< 0	Fehler.

Siehe auch **report4dateFormat**

obj4decimals

Aufruf int obj4decimals(OBJ4 *obj, int numDecimals)

Beschreibung Diese Funktion legt die bei der Ausgabe von numerischen Ausgabeobjekten verwendete Anzahl von Dezimalstellen fest. Un-

genutzte Dezimalstellen werden mit Nullen aufgefüllt. Hat das Ausgabeobjekt als Ergebnis keinen Zahlenwert, zeigt diese Funktion keine Wirkung.

Alle numerischen Ausgabeobjekte, bis auf solche, die mit **obj4-fieldCreate** erzeugt wurden, haben per Voreinstellung keine Dezimalstellen. Ausgabeobjekte numerischer Felder verwenden per Voreinstellung die Anzahl der Dezimalstellen, die auch das Feld aufweist.

Parameter

obj	Ein Zeiger auf das Objekt, für das die Anzahl der Dezimalstellen eingestellt wird.
numDecimals	Anzahl der bei dem Ausgabeobjekt verwendeten Dezimalstellen.

Rückgaben

0	Erfolg.
< 0	Fehler. *obj* ist ungültig.

Siehe auch **report4decimal**

obj4delete

Aufruf void obj4delete(OBJ4 *obj)

Beschreibung Hierbei handelt es sich um eine Universalfunktion, mit der Objekte beliebigen Typs gelöscht werden können. **obj4delete** ermittelt den Typ des Ausgabeobjekts *obj* und ruft die entsprechende „free“-Funktion auf.

Parameter

obj	Ein Zeiger auf das Objekt, das aus dem Report entfernt werden soll.

Siehe auch **report4free**

obj4displayOnce

Aufruf int obj4displayOnce(OBJ4 *obj, char *supprExpr)

Beschreibung Diese Funktion sorgt dafür, daß das angegebene Ausgabeobjekt lediglich bei einer Wertänderung des vorhandenen Ausdrucks ausgegeben wird.

Parameter

obj	Ein Zeiger auf das Objekt, für das die Option „Einmal ausgeben“ eingerichtet wird.
supprExpr	Ein null-terminiertes Zeichenarray, das einen dBASE-Ausdruck enthält, der bestimmt, wann das Ausgabeobjekt ausgegeben wird. Bei Zurücksetzung der Gruppe, in der das Objekt sich befindet, wird der Ausdruck ausgewertet. Hat sich der Wert des Ausdrucks seit der letzten Auswertung verändert, erfolgt die Ausgabe. Ist der Wert gleichgeblieben, wird das Ausgabeobjekt ignoriert.

Rückgaben

0	Erfolg.
< 0	Fehler. *obj* ist ungültig.

obj4displayZero

Aufruf int obj4displayZero(OBJ4 *obj, int displayZero)

Beschreibung Diese Funktion gibt an, ob ein Nullwert für ein numerisches Ausgabeobjekt ausgegeben werden soll oder nicht.

Hat das angegebene Ausgabeobjekt als Ergebnis keinen Zahlenwert, zeigt diese Funktion keine Wirkung.

Parameter

obj	Ein Zeiger auf das Ausgabeobjekt, für das die Option „Null anzeigen“ eingerichtet wird.

	displayZero	Dieses Wahr-/Falsch-Flag kann zwei Werte annehmen: 1 Hat *displayZero* den Wert „1“ (wahr), werden Nullwerte ausgegeben. Wird diese Funktion nicht aufgerufen, geht das Programm von diesem Wert aus. 0 Hat *displayZero* den Wert „0“ (falsch), werden für das angegebene Ausgabeobjekt keine Nullwerte ausgegeben.
Rückgaben	0	Erfolg.
	< 0	Fehler. *obj* ist ungültig.
Siehe auch	**obj4brackets, obj4numericType**	

obj4exprCreate

Aufruf OBJ4 *obj4exprCreate(AREA4 *area, EXPR4 *expr, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein Ausdrucksausgabeobjekt; der verwendete Ausdruck wurde mit der CodeBase-5-Funktion **expr4-parse** erstellt.

Parameter	area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Ausdrucksausgabeobjekt gesetzt werden soll.
	expr	Ein **EXPR4**-Zeiger auf einen analysierten Ausdruck.
	x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Ausdrucksobjekts steht.
	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Ausdrucksobjekts steht.

	width	Die horizontale Ausdehnung des Ausdrucksausgabeobjekts in tausendstel Zoll.
	height	Die vertikale Ausdehnung des Ausdrucksausgabeobjekts in tausendstel Zoll.
Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Ausdrucksausgabeobjekt zurückgegeben.
	0	Fehler. Das Ausdrucksausgabeobjekt konnte nicht angelegt werden.

Siehe auch die CodeBase-5-Funktion **expr4parse, obj4exprFree**

obj4exprFree

Aufruf void obj4exprFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht das angegebene Ausdrucksausgabeobjekt aus dem Report und gibt den mit dem Objekt verbundenen Speicher frei.

Die Funktion gibt durch einen Aufruf der CodeBase-5-Funktion **expr4free** automatisch den Ausdruck frei, auf dem das Ausdrucksausgabeobjekt basiert.

Parameter	obj	Zeiger auf das aus dem Report zu löschende Ausdrucksausgabeobjekt.

Siehe auch die CodeBase-5-Funktion **expr4free, obj4exprCreate, obj4-delete, report4free**

obj4fieldCreate

Aufruf OBJ4 *obj4fieldCreate(AREA4 *area, FIELD4 *field, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein Feldausgabeobjekt für das angegebene Datendateifeld. Bei Feldausgabeobjekten werden automatisch alle Leerzeichen am Ende entfernt, so daß man sie mittig, links- oder rechtsbündig ausrichten kann.

Parameter

area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Feldausgabeobjekt gesetzt werden soll.
field	Ein **FIELD4**-Zeiger auf ein Datendateifeld. Sie können ihn mit der CodeBase-5-Funktion **d4field** ermitteln.
x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Feldobjekts steht.
y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Feldobjekts steht.
width	Die horizontale Ausdehnung des Feldausgabeobjekts in tausendstel Zoll.
height	Die vertikale Ausdehnung des Feldausgabeobjekts in tausendstel Zoll.

Rückgaben

Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Feldausgabeobjekt zurückgegeben.
0	Fehler. Das Feldausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4fieldfree, obj4delete**, die CodeBase-5-Funktion **d4field**

obj4fieldFree

Aufruf void obj4fieldFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein Feldausgabeobjekt aus dem Report und gibt den mit dem Objekt verbundenen Speicher frei. Die Funktion wirkt sich nicht auf das Datendateifeld aus, auf das das Ausgabeobjekt verweist.

Parameter obj — Ein Zeiger auf das (mit **obj4fieldCreate** angelegte) aus dem Report zu löschende Feldausgabeobjekt.

Siehe auch **obj4fieldCreate, obj4delete, report4free**

obj4frameCorners

Aufruf int obj4frameCorners(OBJ4 *obj, int rounded)

Beschreibung Diese Funktion wird bei Rahmenausgabeobjekten eingesetzt, um festzulegen, welcher Eckentyp (runde oder gewinkelte) bei der Objektausgabe verwendet werden soll.

Parameter obj — Ein Zeiger auf das Rahmenausgabeobjekt, für das der Eckentyp festgelegt wird.

rounded Dieser Parameter bestimmt, ob die Ecken des Rahmenobjekts gerundet sind oder nicht. *rounded* kann einen der folgenden Werte annehmen:

- 1 Das Rahmenobjekt wird mit gerundeten Ecken angelegt.
- 0 Das Rahmenobjekt wird mit gewinkelten Ecken (90 Grad) angelegt. Wird diese Funktion nicht aufgerufen, geht das Programm von dieser Einstellung aus.

Rückgaben	>= 0	Die vorhergehende Einstellung von *rounded* wird zurückgegeben.
	< 0	Fehler. *obj* ist ungültig.

Siehe auch **obj4frameCreate, obj4frameFree**

obj4frameCreate

Aufruf OBJ4 *obj4frameCreate(AREA4 *area, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein Rahmenausgabeobjekt im angegebenen Reportbereich. Per Voreinstellung handelt es sich um einen nicht gefüllten Rahmen mit gewinkelten Ecken.

Parameter	area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Rahmenausgabeobjekt gesetzt werden soll.
	x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Rahmenobjekts steht.
	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Rahmenobjekts steht.
	width	Die horizontale Ausdehnung des Rahmenausgabeobjekts in tausendstel Zoll.
	height	Die vertikale Ausdehnung des Rahmenausgabeobjekts in tausendstel Zoll.

Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Rahmenausgabeobjekt zurückgegeben.
	0	Fehler. Das Rahmenausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4frameCorners, obj4frameFill, obj4frameFree, obj4line-Width**

obj4frameFill

Aufruf int obj4frameFill(OBJ4 *obj, int fill)

Beschreibung Mit dieser Funktion wird die Füllung für das angegebene Rahmenausgabeobjekt festgelegt. Hat der Rahmen eine Füllung, stellen die Reportmodul-Ausgabefunktionen diesen in der Farbe des für den Rahmen ausgewählten Styles dar. Enthält der Rahmen keine Füllung (Voreinstellung), wird lediglich die Rahmenlinie ausgegeben.

Parameter

obj — Ein Zeiger auf das Rahmenausgabeobjekt, für das der Füllstatus eingerichtet wird.

fill — Dieser Parameter legt fest, ob der Rahmen bei der Ausgabe eine Füllung erhält oder nicht. *fill* kann die folgenden Werte annehmen:

1 Wird *fill* auf „1“ (wahr) eingestellt, erhält das Rahmenausgabeobjekt eine Füllung.

0 Wird *fill* auf „0“ (falsch) eingestellt, hat das Rahmenausgabeobjekt keine Füllung.

Rückgaben

0 — Die vorhergehende Einstellung von *fill* wird zurückgegeben.

< 0 — Fehler. *obj* oder *fill* sind ungültig.

Siehe auch **obj4frameCreate, obj4frameCorners, obj4lineWidth**

obj4frameFree

Aufruf void obj4frameFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein Rahmenausgabeobjekt aus dem Report und gibt den mit dem Ausgabeobjekt verbundenen Speicher frei.

Parameter

obj	Ein Zeiger auf das zu löschende Rahmenausgabe-objekt.

Siehe auch **obj4frameCreate, obj4delete, report4free**

obj4justify

Aufruf int obj4justify(OBJ4 *obj, int justification)

Beschreibung Diese Funktion legt fest, ob Objekte mit Textausgaben innerhalb der Objektfläche mittig, links- oder rechtsbündig ausgerichtet werden.

Per Voreinstellung werden alle Objekte linksbündig ausgerichtet.

Parameter

obj	Bezeichnet das auszurichtende Objekt.
justification	Beim Parameter *justification* kann es sich um eine der folgenden vordefinierten Konstanten handeln:

justify4left	Der Text für das Ausgabeobjekt wird ab der äußerst links gelegenen Begrenzung des Objekts ausgegeben.
justify4right	Der Text für das Ausgabeobjekt wird ab der äußerst rechts gelegenen Begrenzung des Objekts ausgegeben; dabei beginnt die Ausgabe mit dem letzten Zeichen des Objekts und setzt sich nach links fort.

justify4center	Der Text für das Ausgabeobjekt wird innerhalb der Begrenzung des Objekts zentriert.

Rückgaben

>= 0	Die vorhergehende Einstellung von *justification* wird zurückgegeben.
< 0	Fehler.

Siehe auch **obj4exprCreate, obj4fieldCreate, obj4totalCreate, obj4textCreate**

obj4leadingZero

Aufruf int obj4leadingZero(OBJ4 *obj, int leadingZero)

Beschreibung Mit dieser Funktion wird die Option „Führende Null“ für das angegebene Ausgabeobjekt eingestellt. Hat das Ausgabeobjekt als Ergebnis einen Zahlenwert zwischen 1 und -1, wird diese Option verwendet, um festzulegen, ob dem Dezimalbruch eine Null vorangestellt wird oder nicht.

Führende Null	Keine führende Null
0,33	,33
-0,33	-,33
3,33	3,33

Parameter

obj	Ein Zeiger auf das numerische Ausgabeobjekt, für das die Option „Führende Null“ eingerichtet wird.
leadingZero	Dieser Parameter legt fest, ob bei dem numerischen Ausgabeobjekt eine führende Null verwendet wird oder nicht. *leadingZero* kann die folgenden Werte annehmen:

1 Bei Dezimalbrüchen wird eine führende Null verwendet.

0 Bei Dezimalbrüchen wird keine führende Null verwendet. Wird diese Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben >= 0 Der vorhergehende Wert von *leadingZero* wird zurückgegeben.

< 0 Fehler. *obj* und/oder *leadingZero* sind ungültig.

Siehe auch **obj4displayZero**

obj4lineCreate

Aufruf OBJ4 *obj4lineCreate(AREA4 *area, int vertical, long x, long y, long length)

Beschreibung Mit dieser Funktion werden waagerechte und senkrechte Linienausgabeobjekte von angegebener Länge erzeugt.

Die voreingestellte Linienstärke des Ausgabeobjekts beträgt ein Tausendstel Zoll (1/1 000 Zoll)

Parameter

area Ein Zeiger auf den Bereich, in den das Linienausgabeobjekt gesetzt wird.

vertical Dieser Parameter legt fest, ob es sich um ein senkrechtes bzw. waagerechtes Linienausgabeobjekt handelt. *vertical* kann einen der folgenden Werte annehmen:

1 Beim Linienausgabeobjekt handelt es sich um eine senkrechte Linie.

0 Beim Linienausgabeobjekt handelt es sich um eine waagerechte Linie.

	x	Die waagerechte Koordinate in tausendstel Zoll, an der die Linie beginnt.
	y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der das Linienausgabeobjekt beginnt.
	length	Die Länge des Linienobjekts in tausendstel Zoll.
Rückgaben	Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Linienausgabeobjekt zurückgegeben.
	0	Fehler. Das Linienausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4lineFree, obj4lineWidth**

obj4lineFree

Aufruf void obj4lineFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht ein Linienausgabeobjekt aus dem Report und gibt den mit dem Ausgabeobjekt verbundenen Speicher frei.

Parameter obj Ein Zeiger auf das aus dem Report zu löschende Linienobjekt.

Siehe auch **obj4lineCreate, obj4delete, report4free**

obj4lineWidth

Aufruf int obj4lineWidth(OBJ4 *obj, long width)

Beschreibung Mit dieser Funktion ändern Sie die Linienstärke von Linien- und Rahmenausgabeobjekten.

Parameter	obj	Ein Zeiger auf das Linienausgabeobjekt, für das die Linienstärke neu eingestellt wird.
	width	Die neue Linienstärke des Linienobjekts in tausendstel Zoll.

Rückgaben	0	Die Stärke wurde erfolgreich eingestellt.
	-1	Fehler. *obj* ist ungültig, oder *width* ist eine negative Zahl.

Siehe auch **obj4lineCreate**

obj4lookAhead

Aufruf int obj4lookAhead(OBJ4 *obj, int lookAhead)

Beschreibung Diese Funktion richtet das angegebene Ausgabeobjekt als Vorschauobjekt ein. Bei seiner Ausgabe hat das Objekt den Wert, den es im Gruppenfuß gehabt hätte.

Hinweis: Ein Vorschausummenausgabeobjekt enthält den Wert, den es bei einer Änderung des Summenrücksetzausdrucks gehabt hätte, also nicht unbedingt den im Gruppenfuß.

Parameter	obj	Ein Zeiger auf das Ausgabeobjekt, für das die Vorschau-Option eingerichtet wird.
	lookAhead	Dieser Parameter legt fest, ob es sich bei dem Ausgabeobjekt um ein Vorschauobjekt handelt oder nicht. *lookAhead* kann einen der folgenden Werte annehmen: 1 Das angegebene Objekt ist als Vorschauobjekt eingerichtet. 0 Das Objekt ist nicht als Vorschauobjekt eingerichtet. Wird diese Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben

>= 0	Der vorhergehende Wert von *lookAhead* wird zurückgegeben.
< 0	Fehler. *obj* oder *lookAhead* sind ungültig.

obj4numericType

Aufruf int obj4numericType(OBJ4 *obj, int numericType)

Beschreibung Diese Funktion legt die Formatierung des angegebenen numerischen Ausgabeobjekts fest.

Parameter

obj — Ein Zeiger auf ein Ausgabeobjekt, das als Ergebnis einen Zahlenwert hat.

numericType — Über diesen Parameter wird die Formatierung des numerischen Ausgabeobjekts eingestellt. *numericType* kann eine der folgenden vordefinierten Konstanten sein:

obj4numNumber	Das Ausgabeobjekt wird nicht formatiert.
obj4numExponent	Das Ausgabeobjekt wird in wissenschaftlicher Notation formatiert (d. h., *n.nnnnn e xx*, wobei es sich bei *n* um die Mantisse und bei *x* um den Exponenten handelt.
obj4numCurrency	Das (mit **report4currency** eingestellte) Währungssymbol wird dem Zahlenwert vorangestellt.
obj4numPercent	Bei der Ausgabe wird dem (mit hundert multiplizierten) Zahlenwert unmittelbar das Prozentzeichen nachgestellt.

Rückgaben >= 0 Die vorhergehende Einstellung von *numericType* wird zurückgegeben.

< 0 Fehler. *obj* oder *numericType* sind ungültig.

Siehe auch **report4currency, report4decimal, obj4decimals**

obj4style

Aufruf int obj4style(OBJ4 *obj, STYLE4 *style)

Beschreibung Per Voreinstellung werden alle Ausgabeobjekte mit dem zur Zeit geladenen Style angelegt. Wurde bislang noch kein Style ausgewählt, benutzt das Reportmodul den voreingestellten Style „Normal“.

Diese Funktion verbindet das Ausgabeobjekt mit dem angegebenen Style.

Parameter

obj	Ein Zeiger auf das Ausgabeobjekt, für das der Style eingerichtet wird.
style	Ein Zeiger auf den für das Ausgabeobjekt verwendeten Style.

Rückgaben 0 Erfolg.

< 0 Fehler.

Siehe auch **style4create, style4lookup, style4next**

obj4textCreate

Aufruf OBJ4 *obj4textCreate(AREA4 *area, char *text, long x, long y, long width, long height)

Beschreibung Diese Funktion erzeugt ein statisches Textobjekt.

Parameter

area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Textausgabeobjekt gesetzt werden soll.
text	Ein null-terminiertes Zeichenarray, das den auszugebenden Text enthält. **obj4textCreate** legt eine Kopie von *text* an.
x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Textobjekts steht.
y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Textobjekts steht.
width	Die horizontale Ausdehnung des Textausgabeobjekts in tausendstel Zoll.
height	Die vertikale Ausdehnung des Textausgabeobjekts in tausendstel Zoll.

Rückgaben

Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Textausgabeobjekt zurückgegeben.
0	Fehler. Das Textausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4textFree, obj4delete**

obj4textFree

Aufruf void obj4textFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht das angegebene statische Textausgabeobjekt aus dem Report und gibt den mit dem Ausgabeobjekt verbundenen Speicher frei.

Parameter obj — Ein Zeiger auf das statische Textobjekt, das aus dem Report entfernt werden soll.

Siehe auch **obj4textCreate, obj4delete, report4free**

obj4totalCreate

Aufruf

```
OBJ4 *obj4totalCreate( AREA4 *area, TOTAL4 *total, long x,
                       long y, long width, long height )
```

Beschreibung Diese Funktion erzeugt ein Summenausgabeobjekt unter Verwendung einer mit **total4create** erstellten Summe.

Parameter

area	Dieser **AREA4**-Zeiger bezeichnet den Reportbereich, in den das neue Summenausgabeobjekt gesetzt werden soll.
total	Ein **TOTAL4**-Zeiger auf eine mit **total4create** erstellte Summe.
x	Die waagerechte Koordinate in tausendstel Zoll, an der die linke Seite des Summenobjekts steht.
y	Die senkrechte Koordinate in tausendstel Zoll (vom oberen Rand des Reportbereichs), an der die Oberkante des Summenobjekts steht.
width	Die horizontale Ausdehnung des Summenausgabeobjekts in tausendstel Zoll.
height	Die vertikale Ausdehnung des Summenausgabeobjekts in tausendstel Zoll.

Rückgaben

Nicht-Null	Bei Erfolg wird ein Zeiger auf das neue Summenausgabeobjekt zurückgegeben.
0	Fehler. Das Summenausgabeobjekt konnte nicht angelegt werden.

Siehe auch **obj4totalFree, total4create, total4free, obj4lookAhead**

obj4totalFree

Aufruf void obj4totalFree(OBJ4 *obj)

Beschreibung Diese Funktion löscht das angegebene Summenausgabeobjekt aus dem Report und gibt den mit dem Objekt verbundenen Speicher frei. Darüber hinaus werden alle Ausgabeobjekte bzw. Ausdrücke, die die Summe benutzen, ebenfalls gelöscht. Auch die Summe, auf der das Objekt basiert, wird freigegeben.

Achtung! Diese Funktion kann bei der Löschung von Objekten eine Kettenreaktion auslösen und so einen Report rasch zerstören.

Parameter

obj	Das Summenausgabeobjekt, das aus dem Report entfernt werden soll.

Siehe auch **obj4totalCreate, total4create, total4free, obj4delete, report4-free**

relate4-Funktionen

Mit den beiden hier aufgeführten **relate4**-Funktionen haben Sie die Möglichkeit, eine Relation auf der Platte abzulegen bzw. von der Platte einzulesen. Die Funktionen beschränken sich nicht auf CodeReporter und können daher in jeder CodeBase-5-Anwendung eingesetzt werden.

relate4retrieve

Aufruf RELATE4 *relate4retrieve(CODE4 *cb, char *fileName, int openFiles, char *dataPathName)

Beschreibung Diese Funktion liest eine Relationsdatei ein und baut die mit **relate4save** gespeicherte Relation auf. Beim Laden der Relationsdatei kann man gleichzeitig mit **relate4retrieve** die entsprechenden Datendateien öffnen.

Parameter

cb	Ein Zeiger auf die **CODE4**-Struktur der Anwendung; er dient der Speicherverwaltung und Fehlerbehandlung.
fileName	Ein null-terminiertes Zeichenarray, das den Namen (einschließlich Laufwerk und Verzeichnis) der Relationsdatei enthält. Da stets die Erweiterung .REL verwendet wird, brauchen Sie keinen Dateityp einzugeben.
openFiles	Hat *openFiles* einen wahren Wert (Nicht-Null), versucht **relate4retrieve**, die in der abgespeicherten Relationsdatei genannten Daten-, Index- und Memodateien zu öffnen, falls sie noch nicht geöffnet sind. Kann **relate4retrieve** eine in der Relationsdatei angegebene Datendatei nicht finden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus der Relation

		ausgeklammert und der Versuch unternommen, die nächste Datendatei zu lokalisieren. Hat *openFiles* einen falschen Wert (Null), geht **relate4retrieve** davon aus, daß alle Daten-, Index- und Memodateien bereits geöffnet sind. Kann eine in der Relationsdatei angegebene Datendatei nicht geöffnet werden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus der Relation ausgeklammert, und **relate4-retrieve** setzt den Aufbau der Relation fort.
	dataPathName	Bei diesem Parameter handelt es sich um ein null-terminiertes Zeichenarray, das für die in der Relation gespeicherten Daten-, Index- und Memodateien einen neuen Pfadnamen angibt. Ist *dataPathName* Null, werden die in der Relationsdatei abgelegten Pfade verwendet. Enthält die Datei keine Pfadangaben, versucht **relate4retrieve**, die Dateien im aktuellen Verzeichnis zu öffnen. Wenn Sie *dataPathName* angeben, werden dadurch die in der Relationsdatei gespeicherten Pfade außer Kraft gesetzt.
Rückgaben	Nicht-Null	Die Relation wurde erfolgreich aus der angegebenen Relationsdatei eingelesen.
	Null	Beim Lesen der Relationsdatei oder beim Öffnen der Top-Master-Datendatei hat sich ein Fehler ereignet. Fragen Sie **CODE4.error_code** ab, um den genauen Fehler zu ermitteln.

Siehe auch **relate4save, relate4init, relate4free**

relate4save

Aufruf int relate4save(RELATE4 *relate, char *fileName, int savePathNames)

Beschreibung Mit dieser Funktion wird die angegebene Relation in einer Relationsdatei gespeichert.

Parameter

relate	Ein Zeiger auf die Relation, die in einer Relationsdatei gespeichert werden soll.
fileName	Ein null-terminiertes Zeichenarray, das den Namen (einschließlich Laufwerk und Verzeichnis) der Relationsdatei enthält. Da stets die Erweiterung .REL verwendet wird, brauchen Sie keinen Dateityp einzugeben.
savePathNames	Wenn dieser Parameter einen wahren Wert (nicht-Null) enthält, speichert **relate4save** die in der Relation verwendeten Dateinamen als vollständige Pfadnamen ab. Ist der Wert von *savePathNames* falsch (Null), wird lediglich der reine Dateiname gespeichert.

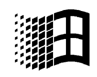
Rückgaben

0	Relationsdatei erfolgreich gespeichert.
r4no_create	Die Relationsdatei konnte nicht angelegt werden. Die Ursache dafür besteht im allgemeinen darin, daß *fileName* bereits vorhanden ist oder die Anwendung keine Lese-/Schreibberechtigung für das angegebene Laufwerk besitzt.
< 0	Fehler.

Siehe auch **relate4retrieve**

report4-Funktionen

Mit den **report4**-Funktionen lassen sich reportweite Einstellungen verändern, z. B. Seitenbreite, Ränder, Währungssymbol, ob die Ausgabe auf dem Bildschirm, dem Drucker usw. erfolgt.



report4caption

Aufruf int report4caption(REPORT4 *report, char *caption)

Beschreibung Mit dieser Funktion wird der Text in der Titelleiste des Reportausgabefensters bei der Bildschirmausgabe verändert.

Parameter

report	Ein Zeiger auf den Report, für den die Titelleiste des Reportausgabefensters eingerichtet wird.
caption	Ein null-terminiertes Zeichenarray, das den Text enthält, der in der Titelleiste des Ausgabefensters erscheinen soll. **report4caption** legt eine Kopie von *caption* an.

Rückgaben

0	Der Reportkopf wurde erfolgreich eingerichtet.
< 0	Fehler.

report4currency

Aufruf int report4currency(REPORT4 *report, char *currency)

Beschreibung Mit dieser Funktion wird der Text festgelegt, der unmittelbar links von solchen numerischen Ausgabeobjekten erscheinen soll, die als Währungswerte formatiert sind.

Parameter report Ein Zeiger auf den Report, für den das Währungssymbol definiert wird.

currency Ein null-terminiertes Zeichenarray, das das Währungssymbol enthält. *currency* kann aus bis zu zehn Zeichen bestehen. **report4currency** legt eine Kopie von *currency* an. Wird diese Funktion nicht aufgerufen, geht das Programm vom Dollarzeichen ($) aus.

Rückgaben 0 Währungssymbol erfolgreich eingerichtet.

< 0 Fehler.

Siehe auch **obj4numericType**

report4dataDo

Aufruf int report4dataDo(REPORT4 *report)

Beschreibung Diese Funktion gibt den Report in eine Datendatei aus, wie in der Schablone für die Reportausgabedatei oder mit den Funktionen **obj4dataFieldSet, report4dataFileSet** und **report4dataGroup** angegeben.

Parameter report Ein Zeiger auf den Report, der in eine Datendatei ausgegeben werden soll.

Rückgaben 0 Erfolg.

< 0 Fehler. *report* ist ungültig oder enthält keine Datendateischablone.

Siehe auch **obj4dataFieldSet, report4dataFileSet, report4dataGroup**

report4dataFileSet

Aufruf int report4dataFileSet(REPORT4 *report, char *destFile)

Beschreibung Mit dieser Funktion wird der Dateiname eingestellt, den die Ausgabedatendatei erhält, wenn die Reportausgabe mit **report4-dataDo** in eine Datendatei erfolgt.

Parameter

report	Ein Zeiger auf den Report, für den der Name der Datendatei eingerichtet wird.
destFile	Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Namen der Datendatei enthält, in die die Reportausgabe erfolgen soll. Werden Laufwerk und/oder Verzeichnis nicht angegeben, geht das Programm vom aktuellen Verzeichnis aus.

Rückgaben

0	Erfolg.
< 0	Fehler. *report* oder *destFile* sind ungültig.

Siehe auch **obj4dataFieldSet, report4dataGroup**

report4dataGroup

Aufruf int report4dataGroup(REPORT4 *report, GROUP4 *group)

Beschreibung Mit dieser Funktion wird die Gruppe bestimmt, deren Rücksetzbedingung in der Ausgabedatendatei einen neuen Datensatz erzeugt. Die Ausgabeobjekte werden mit den Werten in dem erzeugten Datensatz gespeichert, die sie bei der Ausgabe im Fußbereich der Gruppe *group* gehabt hätten.

Parameter

report	Ein Zeiger auf den Report, der mit der Gruppe und der Datendatei verbunden ist.
group	Ein **GROUP4**-Zeiger auf die Gruppe, die Datensätze in der Ausgabedatendatei erzeugt.

Rückgaben 0 Erfolg.

< 0 Fehler. *report* oder *group* sind ungültig.

Siehe auch **report4groupLookup, obj4dataFieldSet, report4dataFileSet**

report4dateFormat

Aufruf int report4dateFormat(REPORT4 *report, char *format)

Beschreibung Diese Funktion richtet das Standard-Datumsformat für den angegebenen Report ein. Alle neuen Ausgabeobjekte, die als Ergebnis einen Datumswert haben, benutzen dieses Format bei der Ausgabe. Beim ersten Anlegen des Reports wird der Wert des Elements **CODE4.date_format** als Standard-Datumsformat des Reports gespeichert.

Parameter report Ein Zeiger auf den Report, für den das Datumsformat eingestellt wird.

format Ein null-terminiertes Zeichenarray, das das Standard-Datumsformat enthält. Diese Zeichenkette sollte aus den Schablonenformatierzeichen bestehen („D“, „M“, „C“, „Y“). **report4dateFormat** legt eine Kopie von *format* an, so daß *format* auf temporären Speicher zeigen kann.

Rückgaben 0 Erfolg.

< 0 Fehler. *report* ungültig oder nicht genügend Speicher, um *format* zu kopieren.

Siehe auch **obj4dateFormat**

report4decimal

Aufruf int report4decimal(REPORT4 *report, char decimalChar)

Beschreibung Mit dieser Funktion wird das Zeichen bestimmt, das bei numerischen Ausgabeobjekten die ganzen Zahlen von den Bruchwerten trennt.

Parameter

report	Ein Zeiger auf den Report, bei dem das Dezimalzeichen verwendet wird.
decimalChar	Das als Dezimalzeichen zu verwendende Zeichen. Wird diese Funktion nicht aufgerufen, geht das Programm vom Dezimalpunkt (.) aus.

Rückgaben

0	Erfolg.
< 0	Fehler. *report* ist ungültig.

Siehe auch **obj4decimals**

report4do

Aufruf int report4do(REPORT4 *report)

Beschreibung Mit dieser High-Level-Funktion wird der Report auf das ausgewählte Gerät ausgegeben.

Bei der Ausgabe unter Windows deaktiviert **report4do** das (mit **report4parent** bezeichnete) Elternfenster des Reports und erzeugt ein Ausgabefenster, das den Report anzeigt. Nach Abschluß des Vorgangs sendet das Ausgabefenster eine CRM_REPORTCLOSED-Meldung an das Elternfenster.

Achtung! Um diese Funktion einsetzen zu können, müssen Sie zunächst **report4parent** aufrufen. Erfolgt dieser Aufruf nicht, kann das zu unvorhersehbaren Ergebnissen führen.

Wird der Report in einer Non-Windows-Anwendung ausgegeben, schickt **report4do** den Report an das mit **report4output** angegebene Gerät und springt nach Ausführung des Reports zurück.

Parameter report — Bezeichnet den auszugebenden Report.

Rückgaben

0	Der Report wurde erfolgreich ausgegeben. Unter Windows wird dieser Wert augenblicklich zurückgegeben, auch wenn die Ausgabe des Reports im Ausgabefenster noch nicht ganz abgeschlossen ist.
r4terminate	Es konnte keine Relation hergestellt werden, und bei der mit **relate4error_action** angegebenen Fehlerbehandlung handelte es sich um **relate4-terminate**.
< 0	Fehler.

Siehe auch **report4toScreen, report4parent, report4printerSelect, report4printerSet, report4output**

report4free

Aufruf

```
void report4free( REPORT4 *report, int freeRelate,
                                    int closeFiles )
```

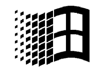
Beschreibung Diese Funktion gibt den gesamten mit dem Report verbundenen Speicher, einschließlich den aller Ausgabeobjekte, Gruppen und Bereiche, frei.

Achtung! Bei einer Windows-Anwendung sollte diese Funktion erst aufgerufen werden, nachdem das Elternfenster des Reports eine CRM_REPORTCLOSED-Meldung erhalten hat. Wird **report4-free** unter Windows unmittelbar nach **report4do** aufgerufen, kann das zu unvorhersehbaren Ergebnissen führen.

Parameter

report	Der Report, der aus dem Speicher gelöscht werden soll.
freeRelate	Hat dieser Parameter einen wahren Wert (nicht-Null), wird der mit der Relation des Reports verbundene Speicher automatisch freigegeben. Wird ein falscher Wert (Null) übergeben, wirkt sich das nicht auf die Relation aus.
closeFiles	Dieser Parameter schließt, wenn er einen wahren Wert (nicht-Null) enthält, automatisch die im Report genannten Daten-, Index- und Memodateien. Ist *freeRelate* falsch (null), wird die Einstellung von *closeFiles* ignoriert.

Siehe auch **report4pageFree** sowie **relate4free** im CodeBase-5-Referenzhandbuch



report4generatePage

Aufruf int report4generatePage(REPORT4 *report, HDC hDC)

Beschreibung Diese Low-Level-Funktion wird dazu verwendet, die nächste Reportseite in einen Windows-Gerätekontext (z. B. einen Bitmap-Gerätekontext oder einen Druckergerätekontext) zu kopieren. Der vorhergehende Gerätekontext wird bei dieser Funktion nicht gelöscht.

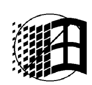
Soll der Report mit Hilfe dieser Funktion auf einem Drucker ausgegeben werden, ist der Programmierer dafür verantwortlich, daß die Codes SETABORTPROC (falls gewünscht), STARTDOC, NEWFRAME und ENDDOC mit der Windows-Abbruchfunktion auf das Gerät geschickt werden.

Soll der Report mit Hilfe dieser Funktion in eine Bitmap (zwecks Anzeige in einem Fenster, Speicherung auf der Platte usw.)

ausgegeben werden, muß der Programmierer einen Speichergerätekontext mit einer Bitmap erzeugen, deren Größe zum Speichern einer Reportseite ausreicht.

Parameter	report	Ein Zeiger auf den Report, aus dem eine Seite eingelesen wird.
	hDC	Ein Handle auf einen gültigen Microsoft-Windows-Gerätekontext, in den die nächste Seite des Reports kopiert wird. Der Programmierer ist dafür verantwortlich, *hDC* freizugeben, sobald es nicht mehr benötigt wird.
Rückgaben	0	Neue Seite erfolgreich in *hDC* gespeichert.
	2	Der Report enthält keine weiteren Seiten. *hDC* ist unverändert.
	< 0	Fehler. *report* und/oder *hDC* sind ungültig.

Siehe auch **report4init**



report4generatePage

Aufruf int report4generatePage(REPORT4 *report)

Beschreibung Mit dieser Low-Level-Funktion wird eine interne Struktur erzeugt, die Informationen über die nächste Seite eines Reports enthält. Mit Hilfe der Funktionen **report4pageObjFirst** und **report4pageObjNext** lassen sich die Werte der Ausgabeobjekte auf der Reportseite abrufen.

Parameter	report	Ein Zeiger auf den Report, für den die nächste Seite eingelesen wird.

Rückgaben

0	Die neue Seite wurde erfolgreich im internen Puffer abgelegt.
2	Der Report enthält keine weiteren Seiten.
< 0	Fehler. *report* ist ungültig.

Siehe auch **report4pageObjFirst, report4pageObjNext, report4init**

report4groupFirst

Aufruf GROUP4 *report4groupFirst(REPORT4 *report)

Beschreibung **report4groupFirst** gibt einen Zeiger auf die innerste Gruppe zurück; diese Gruppe wurde als erste Gruppe des Report erzeugt.

In Verbindung mit **report4groupNext** dient diese Funktion dazu, die Gruppen eines Reports zu durchlaufen.

Parameter

report	Ein Zeiger auf den Report, aus dem die erste Gruppe abgerufen wird.

Rückgaben

0	Für den angegebenen Report wurde noch keine Gruppe erzeugt.
Nicht-Null	Ein Zeiger auf die innerste Gruppe des Reports.

Siehe auch **report4groupNext, report4numGroups**

report4groupLast

Aufruf GROUP4 *report4groupLast(REPORT4 *report)

Beschreibung **report4groupLast** gibt einen Zeiger auf die äußerste Gruppe zurück; diese Gruppe wurde als letzte des Report erzeugt.

In Verbindung mit **report4groupPrev** dient diese Funktion dazu, die Gruppen eines Reports zu durchlaufen.

Parameter	report	Ein Zeiger auf den Report, aus dem die letzte Gruppe abgerufen wird.
Rückgaben	0	Für den angegebenen Report wurde noch keine Gruppe erzeugt.
	Nicht-Null	Ein Zeiger auf die äußerste Gruppe des Reports.

Siehe auch **report4groupPrev, report4numGroups**

report4groupLookup

Aufruf GROUP4 *report4groupLookup(REPORT4 *report, char *name)

Beschreibung Mit dieser Funktion erhalten sie einen **GROUP4**-Zeiger auf die angegebene Gruppe.

Parameter	report	Ein Zeiger auf den Report, zu dem die Gruppe gehört.
	name	Ein null-terminiertes Zeichenarray, das den Namen der Gruppe enthält, für die der **GROUP4**-Zeiger erforderlich ist.
Rückgaben	Nicht-Null	Ein Zeiger auf die angegebene Gruppe wird zurückgegeben.
	0	*name* stimmt mit keiner Gruppenbezeichnung im angegebenen Report überein.

Siehe auch **group4create**

report4groupNext

Aufruf GROUP4 *report4groupNext(REPORT4 *report,
GROUP4 *group)

Beschreibung Mit dieser Funktion erhält man einen **GROUP4**-Zeiger auf die unmittelbar nach der angegebenen Gruppe erzeugte Gruppe.

In Verbindung mit **group4first** dient die Funktion dazu, die Gruppen eines Reports zu durchlaufen.

Parameter

report	Ein Zeiger auf den Report, zu dem die Gruppe gehört.
group	Ein Zeiger auf eine innere Gruppe, über die die nächste weiter außen gelegene Gruppe ermittelt wird.

Rückgaben

0	Der Report enthält keine weiteren Gruppen; *group* ist die äußerste Gruppe.
Nicht-Null	Ein Zeiger auf die nächste Gruppe wird zurückgegeben.

Siehe auch **group4first, report4numGroups**

report4groupPrev

Aufruf GROUP4 *report4groupPrev(REPORT4 *report,
GROUP4 *group)

Beschreibung Mit dieser Funktion erhält man einen **GROUP4**-Zeiger auf die unmittelbar vor der angegebenen Gruppe erzeugte Gruppe.

In Verbindung mit **report4groupLast** dient die Funktion dazu, die Gruppen eines Reports zu durchlaufen.

Parameter	report	Ein Zeiger auf den Report, zu dem die Gruppe gehört.
	group	Ein Zeiger auf eine äußere Gruppe, über die die innere Gruppe ermittelt wird, die vorher angelegt wurde.
Rückgaben	0	Der Report enthält keine weiteren Gruppen; *group* ist die innerste Gruppe.
	Nicht-Null	Ein Zeiger auf die vorhergehende Gruppe wird zurückgegeben.

Siehe auch **report4groupLast, report4numGroups**

report4hardResets

Aufruf int report4hardResets(REPORT4 *report, int hardResets)

Beschreibung Anhand dieser Funktion wird die Methode festgelegt, wie Gruppen mit der Option „Seite zurücksetzen" neue Seiten beginnen.

Parameter	report	Ein Zeiger auf den Report, für den das Flag „Feste Rücksetzung" gesetzt wird.
	hardResets	Mit diesem Parameter wird festgelegt, auf welche Art und Weise Seitenzurücksetzungen erfolgen. *hardResets* kann einen der folgenden Werte annehmen:

1 Vor Ausgabe einer Seite mit dem Flag „Seite zurücksetzen" immer eine neue Seite anlegen.

0 Für eine Gruppe mit einem Flag „Seite zurücksetzen" lediglich dann eine neue Seite generieren, wenn die Gruppe aufgrund einer Änderung ihrer eigenen Rücksetzbedingung zurückgesetzt wird. Wird die Gruppe aufgrund der

Rücksetzung einer höheren Gruppe zurückgesetzt, wird keine neue Seite generiert. Wird diese Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben >= 0 Der vorhergehende Wert von *hardResets* wird zurückgegeben.

< 0 Fehler. *report* und/oder *hardResets* sind ungültig.

Siehe auch **group4resetPage**

report4init

Aufruf REPORT4 *report4init(RELATE4 *relate)

Beschreibung Diese Funktion initialisiert eine Reportstruktur mit voreingestellten Werten und gibt einen Reportzeiger zurück, der zusammen mit den übrigen Reportmodulfunktionen eingesetzt werden kann.

Die Funktion wird automatisch von **report4retrieve** aufgerufen.

Nach der Ausführung des Reports sollten Sie **report4free** aufrufen, um den mit dem Report verbundenen Speicher wieder freizugeben.

report4init richtet folgende Standardwerte ein:

Ränder	links: 0,64 cm, rechts: 0,64 cm, oben: 0 cm, unten: 0 cm
Seitenformat	21,59 cm x 27,94 cm
Dezimalpunkt	.
Zeichen n. Tausend	,
Währungssymbol	$
Style-Voreinstellung	„Normal“ (Windows MS Serif 10 Pkt., Non-Windows: Keine Steuerzeichen)
Titel-/Schlußgruppe	Größe null
Seitenkopf-/-fußgruppe	Größe null

Parameter relate Die Relation, auf der der Report basiert.

Rückgaben Nicht-Null Der Report wurde erfolgreich initialisiert, und ein Zeiger auf die Reportstruktur wird zurückgegeben.

Null Fehler. Der Report konnte nicht initialisiert werden. Einzelheiten entnehmen Sie bitte der Einstellung von **CODE4.error_code**.

Siehe auch **report4retrieve, relate4retrieve, report4do, report4free**

report4margins

Aufruf int report4margins(REPORT4 *report, long left, long right, long top, long bottom, int unitType)

Beschreibung Mit dieser Funktion werden die Standard-Randeinstellungen des Reports verändert.

Hinweis: Einige Ausgabegeräte, z. B. Laserdrucker, haben einen hardwaremäßig eingestellten Randbereich, der nicht bedruckbar ist. **report4margins** prüft diese Tatsache ab und läßt es nicht zu, daß die physikalischen Ränder des Geräts nicht eingehalten werden.

Parameter report Ein Zeiger auf den Report, bei dem die Ränder eingestellt werden.

left Breite des linken Randes in den Schritten der eingestellten Maßeinheit.

right Breite des rechten Randes in den Schritten der eingestellten Maßeinheit.

top Breite des oberen Randes in den Schritten der eingestellten Maßeinheit.

	bottom	Breite des unteren Randes in den Schritten der eingestellten Maßeinheit.
	unitType	Maßeinheit für die oben genannten Randeinstellungen. Bei graphischen Benutzerschnittstellen lassen sich dabei recht gut Angaben in tausendstel Zoll verwenden. Bei zeichenbezogenen Schnittstellen eignen sich Angaben in Zeichen oft besser. *unitType* kann einen der folgenden Werte annehmen: 1 Die Maßeinheiten werden in Zeichen angegeben. 0 Die Maßeinheiten werden in tausendstel Zoll angegeben.

Hinweis: Ist *unitType* auf Zeichen eingestellt, geht diese Funktion von zehn Zeichen pro Zoll (cpi) und sechs Zeilen pro Zoll (lpi) aus.

Rückgaben
- 0 — Die Ränder wurden erfolgreich eingestellt.
- < 0 — Fehler.

Siehe auch **report4pageSize**

report4numGroups

Aufruf

```
int report4numGroups( REPORT4 *report )
```

Beschreibung Diese Funktion gibt die aktuelle Anzahl der im Report enthaltenen Gruppen zurück. Das ist beim Durchlaufen der Gruppen in einem Report nützlich.

Parameter

report	Dieser **REPORT4**-Zeiger bezeichnet den Report, bei dem die Anzahl der Gruppen ermittelt werden soll.

Rückgaben	0	Der angegebene Report enthält keine Gruppen.
	> 0	Anzahl der im Report enthaltenen Gruppen.
	< 0	Ein Fehler ist aufgetreten.

Siehe auch **report4groupNext, report4groupPrev, group4first, report4groupLast**

report4numStyles

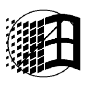
Aufruf int report4numStyles(REPORT4 *report)

Beschreibung Diese Funktion gibt die aktuelle Anzahl der im Report enthaltenen Styles zurück. Das ist beim Durchlaufen der Styles in einem Report nützlich.

Parameter report Dieser **REPORT4**-Zeiger bezeichnet den Report, bei dem die Anzahl der Styles ermittelt werden soll.

Rückgaben	> 0	Anzahl der im Report enthaltenen Styles. Jeder Report enthält zumindest einen Style.
	< 0	Ein Fehler ist aufgetreten.

Siehe auch **report4styleFirst, report4styleNext, report4styleLast, report4stylePrev**



report4output

Aufruf int report4output(REPORT4 *report, int outputHandle, int useStyles)

Beschreibung Der Aufruf dieser Funktion erfolgt vor der Erzeugung des Reports und weist **report4do** an, den Report auf ein System-Handle wie „standard out“, „standard print“ oder in eine offene Datei zu schreiben. Wird diese Funktion nicht aufgerufen, erfolgt die Reportausgabe über **report4do** auf dem „standard out“-Bildschirm.

report4do vollzieht die Reportausgabe über die Standard-Funktion write() der C-Bibliothek, die ein System-Handle zur Ausgabe von Text verwendet. Dabei übergibt **report4do** *output-Handle* an write().

Diese Funktion wird eingesetzt, wenn ein Report in eine Datei ausgegeben wird.

Parameter

report — Der Report, für den ein Ausgabeziel eingerichtet wird.

outputHandle — Ein Standard-C-System-Handle, wie es von den C-Funktionen open() und sopen() zurückgegeben wird. Andere bei der Programmiersprache C vordefinierte Handles sind:

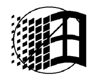
- 1 „standard out“; per Voreinstellung ist das der Bildschirm.
- 4 „standard print“. Bei PCs handelt es sich dabei gewöhnlich um den Drucker an LPT1.

useStyles — Mit diesem Parameter wird festgelegt, ob die von **report4do** ausgegebenen Daten die Druckersteuersequenzen, die in den Seiten-Layouts definiert sind, enthalten sollen oder nicht. *useStyles* kann einen der folgenden Werte annehmen:

- 1 Die Ein- und Ausleitungssequenzen des Druckers werden in das angegebene Handle ausgegeben.
- 0 Die Druckersteuersequenzen werden ignoriert, und es wird lediglich der Text für die Ausgabeobjekte ausgegeben.

Rückgaben 0 Einstellungen erfolgreich vorgenommen.
< 0 Fehler. *report* ist ungültig.

Siehe auch **report4do, report4toScreen**



report4pageFree

Aufruf int report4pageFree(REPORT4 *report)

Beschreibung Diese Low-Level-Funktion dient dazu, den Speicher intern freizugeben, der mit der Darstellung einer Ausgabeseite verbunden ist. Die Funktion wird am Ende des Reports automatisch von **report4do** aufgerufen.

Parameter report Ein Zeiger auf den Report, dessen Ausgabeseite freigegeben wird.

Rückgaben 0 Seite erfolgreich freigegeben.
< 0 Fehler.

Siehe auch **report4free, report4generatePage**

report4pageHeaderFooter

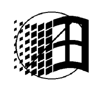
Aufruf GROUP4 *report4pageHeaderFooter(REPORT4 *report)

Beschreibung Diese Funktion gibt einen **GROUP4**-Zeiger auf die Seitenkopf- und -fußgruppe des Reports zurück. Die zurückgegebene Gruppe, die automatisch von **report4init** erzeugt wird, kann nicht gelöscht werden.

Parameter report Ein Zeiger auf den Report, der die gewünschte Seitenkopf-/-fußgruppe enthält.

Rückgaben Ein **GROUP4**-Zeiger auf die Seitenkopf-/-fußgruppe.

Siehe auch **area4create, report4init**



report4pageInit

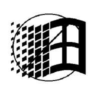
Aufruf int report4pageInit(REPORT4 *report)

Beschreibung Mit dieser Low-Level-Funktion wird ein interner Seitenpuffer für die Ausgabe des Reports erzeugt. Die Funktion wird automatisch von **report4do** aufgerufen.

Parameter report Ein Zeiger auf den Report, für den der interne Seitenpuffer erzeugt wird.

Rückgaben 0 Der Seitenpuffer wurde erfolgreich angelegt.
< 0 Fehler.

Siehe auch **report4pageFree, report4do**



report4pageMarginsGet

Aufruf int report4marginsGet(REPORT4 *report, long *left, long *right, long *top, long *bottom)

Beschreibung Mit dieser Funktion werden die für den Report eingestellten Ränder abgerufen. Alle Ränder werden in Schritten von 1/1 000 Zoll eingelesen.

Parameter

report	Ein Zeiger auf den Report, dessen Randeinstellungen abgerufen werden.
left	Ein Zeiger auf eine **(long)**-Variable, die die Breite des linken Randes enthält.
right	Ein Zeiger auf eine **(long)**-Variable, die die Breite des rechten Randes enthält.
top	Ein Zeiger auf eine **(long)**-Variable, die die Breite des oberen Randes enthält.
bottom	Ein Zeiger auf eine **(long)**-Variable, die die Breite des unteren Randes enthält.

Rückgaben

0	Randeinstellungen erfolgreich eingelesen.
< 0	Fehler. *report* ist ungültig.

Siehe auch **report4margins**

report4pageObjFirst

Aufruf OBJECT4 *report4pageObjFirst(REPORT4 *report)

Beschreibung Diese Low-Level-Funktion dient dazu, eine interne Darstellung des ersten ausgewerteten Ausgabeobjekts auf der aktuellen Seite des Reports einzulesen.

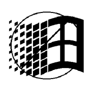
Parameter

report	Ein Zeiger auf den Report, aus dem das erste Objekt der aktuellen Ausgabeseite eingelesen wird.

Rückgaben

>= 0	Ein **OBJECT4**-Zeiger für das ausgewertete erste Objekt der aktuellen Ausgabeseite wird zurückgegeben.
< 0	Die aktuelle Seite enthält keine Objekte.

Siehe auch **report4pageObjNext**



report4pageObjNext

Aufruf OBJECT4 *report4pageObjNext(REPORT4 *report)

Beschreibung Diese Low-Level-Funktion dient dazu, das nächste ausgewertete Ausgabeobjekt auf der aktuellen Seite des Reports einzulesen.

Parameter report — Ein Zeiger auf den Report, aus dem das nächste Objekt der aktuellen Ausgabeseite eingelesen wird.

Rückgaben

>= 0 — Ein **OBJECT4**-Zeiger für das ausgewertete nächste Objekt der aktuellen Ausgabeseite wird zurückgegeben.

< 0 — Die aktuelle Seite enthält keine Objekte.

Siehe auch **report4pageObjFirst**

report4pageSize

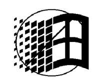
Aufruf int report4pageSize(REPORT4 *report, long height, long width, int unitType)

Beschreibung Mit dieser Funktion werden Breite und Höhe der Reportseite eingestellt. Wird diese Funktion innerhalb einer Windows-Anwendung nicht aufgerufen, wird das aktuelle Seitenformat des ausgewählten Druckers verwendet. Wenn die Funktion in einer Non-Windows-Anwendung nicht aufgerufen wird, wird das voreingestellte Seitenformat von 25 x 80 Zeichen verwendet.

Parameter report — Ein Zeiger auf den Report, für den das Seitenformat eingestellt wird.

	height	Die Höhe der Ausgabeseite in der angegebenen Maßeinheit.
	width	Die Breite der Ausgabeseite in der angegebenen Maßeinheit.
	unitType	Mit diesem Parameter wird die Maßeinheit von *height* und *width* festgelegt. *unitType* kann einen der folgenden Werte annehmen: 1 *height* und *width* werden in Zeichen angegeben. 0 *height* und *width* werden in tausendstel Zoll angegeben.
Rückgaben	0	Das Seitenformat wurde erfolgreich eingerichtet.
	< 0	Fehler.

Siehe auch **report4margins, report4printerSelect**



report4pageSizeGet

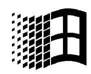
Aufruf int report4pageSizeGet(REPORT4 *report, long *width, long *height)

Beschreibung Diese Funktion fragt das aktuelle Format der Reportseite (in tausendstel Zoll) ab.

Parameter	report	Ein Zeiger auf den Report, bei dem das Seitenformat abgefragt wird.
	width	Ein Zeiger auf eine **(long)**-Variable, in der die Breite der Seite gespeichert werden soll.
	height	Ein Zeiger auf eine **(long)**-Variable, in der die Höhe der Seite gespeichert werden soll.

Rückgaben 0 Das Seitenformat wurde erfolgreich abgefragt.
< 0 Fehler. *report* ist ungültig.

Siehe auch **report4pageSize, report4marginsGet**



report4parent

Aufruf int report4parent(REPORT4 *report, HWND parent)

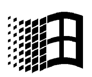
Beschreibung Diese Funktion legt das für die Reportausgabe zu verwendende Elternfenster fest. Bei der Reportausgabe deaktiviert **report4do** das *parent*-Fenster; nachdem das Reportausgabefenster geschlossen wurde, erhält das Elternfenster eine CRM_REPORTDONE-Meldung.

Parameter

report	Ein Zeiger auf den Report, für den das Elternfenster eingerichtet wird.
parent	Ein Microsoft-Windows-Handle, das der Reportausgabe dient.

Achtung! Diese Funktion muß vor **report4do** aufgerufen werden. Geschieht das nicht, kann das zu unvorhersehbaren Ergebnissen führen.

Siehe auch **report4do, report4free**



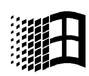
report4printerSelect

Aufruf void report4printerSelect(REPORT4 *report)

Beschreibung Diese Funktion ruft die Dialogbox „Druckereinrichtung“ auf, um einen Drucker für den Report auszuwählen. Damit die Funktion einwandfrei arbeitet, muß die Anwendung unter Microsoft Windows 3.1 (oder höher) laufen, oder die dynamische Link-Bibliothek COMMDLG.DLL muß sich im Systempfad befinden.

Parameter

report	Ein Zeiger auf den Report, der für den ausgewählten Drucker konfiguriert ist.



report4printerDC

Aufruf HDC report4printerDC(REPORT4 *report, HDC hDC)

Beschreibung Mit dieser Funktion wird ein Handle auf einen Druckergerätekontext angegeben, an den die Reportausgabe erfolgt.
Diese Low-Level-Funktion ist dann nützlich, wenn das Handle auf den Druckergerätekontext über Standard-Windows-Funktionsaufrufe ermittelt wird. Um das Reportausgabegerät interaktiv auszuwählen, ist die Funktion **report4printerSelect** vorgesehen.

Dieses Handle ist ein vorab definierter Gerätekontext (falls eingerichtet) oder null, falls kein Gerätekontext eingerichtet wurde. Es liegt in der Verantwortung des Programmierers, den zurückgegebenen Gerätekontext wieder freizugeben.

Hinweis: Der angegebene Druckergerätekontext wird von **report4do** nicht verwendet, falls die Ausgabe mit **report4toScreen** auf dem Bildschirm erfolgt.

Parameter

report	Ein Zeiger auf den Report, der für den angegebenen Druckergerätekontext konfiguriert ist.
hDC	Ein Handle auf den Druckergerätekontext, in den die Reportausgabe gestellt werden soll.

Siehe auch **report4toScreen, report4printerSelect**

report4querySet

Aufruf int report4querySet(REPORT4 *report, char *queryExpr)

Beschreibung Diese Funktion stellt eine Abfrage für das Relationsset ein. Dabei wird der Ausdruck *queryExpr* für jeden zusammengesetzten Datensatz ausgewertet. Ist der Ausdruck wahr, geht der Datensatz in den Report ein. Ist das Ergebnis von *queryExpr* falsch, wird der Datensatz ignoriert.

Achtung! Diese Funktion überschreibt alle mit **relate4query_set** eingerichteten Abfrageausdrücke.

Parameter

report	Ein Zeiger auf den Report, für den die Abfrage eingerichtet wird.
queryExpr	Ein logischer dBASE-Ausdruck, der Datensätze der zusammengesetzten Datendatei filtert. Ist *queryExpr* Null, werden alle Datensätze der zusammengesetzten Datendatei im Report verwendet.

Hinweis: Im Abfrageausdruck vorhandene Feldnamen müssen mit Datendateikennzeichner verwendet werden, z. B. `"DBF->NAME = 'SCHMIDT'"`.

Rückgaben

0	Die Abfrage wurde erfolgreich eingerichtet.
< 0	Fehler.

Siehe auch **relate4query_set, relate4sort_set, report4sortSet**

report4retrieve

Aufruf REPORT4 *report4retrieve(CODE4 *cb, char *fileName,
int openFiles, char *dataPath)

Beschreibung Diese Funktion liest eine Reportdatei von der Platte ein und baut die entsprechende **REPORT4**-Struktur auf. Parallel dazu wird im Hintergrund ein Relationsset mit der entsprechenden **RELATE4**-Struktur erzeugt.

Hinweis: Reportdateien, die mit CodeReporter und/oder **report4save** angelegt wurden, sind nicht unbedingt auf ein anderes Betriebssystem portierbar. Soll der Report auf einer anderen Plattform verwendet werden, muß er möglicherweise mit von CodeReporter generiertem Quellcode gelinkt werden.

Parameter

cb — Ein Zeiger auf die **CODE4**-Struktur der Anwendung; er dient der Speicherverwaltung und der Fehlerbehandlung.

fileName — Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Namen der Reportdatei enthält. Geben Sie keine Erweiterung am Dateinamen an, geht **report4retrieve** von der Erweiterung .REP aus.

openFiles — Hat *openFiles* einen wahren Wert (Nicht-Null), versucht **report4retrieve**, die im Report genannten Datendateien zu öffnen, falls sie noch nicht geöffnet sind. Kann eine angegebene Datendatei nicht lokalisiert werden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus dem Report ausgeklammert. Alle Ausgabeobjekte und/ oder Ausdrücke, die sich auf die fehlenden Datendateien beziehen, werden automatsich aus dem Report entfernt.

		Hat *openFiles* einen falschen Wert (Null), geht das Programm davon aus, daß alle Dateien bereits geöffnet sind.
	dataPath	Ist *dataPath* Null, benutzt **report4retrieve** zur Lokalisierung der Datendateien die in der Reportdatei gespeicherten Pfade. Wenn die Datei keine Pfadnamen enthält, geht **report4retrieve** davon aus, daß sich die Datendateien im aktuellen Verzeichnis befinden. Ist *dataPath* nicht Null, enthält es als null-terminiertes Zeichenarray Laufwerk und/oder Verzeichnis, in dem sich die Datendateien des Reports befinden. Das Verzeichnis *dataPath* setzt alle innerhalb des Reports gespeicherten Pfade außer Kraft.
Rückgaben	Nicht-Null	Der Report wurde erfolgreich geladen. Der zurückgegebene **REPORT4**-Zeiger kann auch von anderen Funktionen des Reportmoduls benutzt werden.
	0	Fehler. Der Report konnte nicht geladen werden. Das liegt möglicherweise daran, daß die Top-Master-Datendatei nicht lokalisiert oder nicht genug Speicher für den Report zugewiesen werden konnte.

Siehe auch **relate4retrieve**

report4save

Aufruf

```
int report4save( REPORT4 *report, char *fileName,
                                        int savePaths )
```

Beschreibung Diese Funktion legt einen Report in einer variabel codierten Reportdatei ab, die entweder mit einer CodeReporter-Anwen-

dung bzw. der Funktion **report4retrieve** wieder eingelesen werden kann.

Hinweis: **report4save** wirkt sich in keinster Weise auf den im Speicher befindlichen Report aus. Die Funktion kann ohne schädliche Wirkung vor oder nach **report4do** aufgerufen werden.

Achtung! Wird ein Report mit Grafikausgabeobjekten in eine Non-Windows-Anwendung geladen und mit **report4save** abgespeichert, werden die Grafikausgabeobjekte in der neuen Reportdatei nicht mitgesichert.

Parameter

report	Ein Zeiger auf den Report, der auf der Platte abgelegt werden soll.
fileName	Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Namen der Datei enthält, unter denen der Report gespeichert werden soll. Sie können eine Erweiterung am Dateinamen angeben; geben Sie keine an, wird die voreingestellte Erweiterung .REP an den Dateinamen angehängt.
savePaths	Enthält *savePaths* einen wahren Wert (nicht-Null), legt **report4save** für jede im Report genannte Datei Laufwerk und Pfad in der Reportdatei ab. Enthält *savePaths* einen falschen Wert (Null), werden lediglich die im Report vorhandenen Dateinamen gespeichert.

Rückgaben

0	Der Report wurde erfolgreich in die angegebene Datei geschrieben.
< 0	Fehler. *report* ist ungültig, oder die angegebene Datei konnte nicht angelegt werden.

Siehe auch **report4retrieve**

report4separator

Aufruf int report4separator(REPORT4 *report, char separator)

Beschreibung Mit dieser Funktion wird das Zeichen angegeben, das zwischen Hundertern und Tausendern, Tausendern und Millionen usw. steht.

Parameter

report	Ein Zeiger auf den Report, für den das Ziffern-trennzeichen gelten soll.
separator	Das als Trennzeichen verwendete Zeichen. Wünschen Sie kein Trennzeichen, übergeben Sie für *separator* eine Null (0). Wird diese Funktion nicht aufgerufen, wird als voreingestelltes Trennzeichen das Komma (,) benutzt.

Rückgaben

0	Trennzeichen erfolgreich eingerichtet.
< 0	Fehler. *report* ist ungültig.

Siehe auch **obj4numericType**

report4sortSet

Aufruf int report4sortSet(REPORT4 *report, char *sortExpr)

Beschreibung Diese Funktion legt die Sortierfolge fest, in der die zusammengesetzten Datensätze des Reports eingelesen werden.

Achtung! Diese Funktion überschreibt alle mit **relate4sort_set** eingerichteten Sortierausdrücke.

Parameter

report	Ein Zeiger auf den Report, für den die Sortierfolge gelten soll.

sortExpr — Ein null-terminiertes Zeichenarray, das den dBASE-Ausdruck enthält, nach dem die zusammengesetzte Datendatei sortiert wird. Dieser Ausdruck kann als Ergebnis einen Zeichen-, Datums- oder einen numerischen Wert haben.

Hinweis: Im Sortierausdruck vorhandene Feldnamen müssen mit Datendateikennzeichner verwendet werden. `"DBF->NAME"` ist ein Beispiel für einen gültigen Sortierausdruck.

Rückgaben

0 — Erfolg.

< 0 — Fehler, oder *report* ist ungültig.

Siehe auch **relate4sort_set, report4querySet**

report4styleFirst

Aufruf STYLE4 *report4styleFirst(REPORT4 *report)

Beschreibung Diese Funktion gibt einen Zeiger auf den ersten für den Report definierten Style zurück. In Verbindung mit **report4styleNext** dient die Funktion dazu, die Styles eines Reports zu durchlaufen.

Parameter report — Ein Zeiger auf den Report, der die Styles enthält.

Rückgaben Ein Zeiger auf den ersten für den Report definierten Style wird zurückgegeben.

Siehe auch **report4styleNext, report4styleLast**

report4styleLast

Aufruf	STYLE4 *report4styleLast(REPORT4 *report)	
Beschreibung	Diese Funktion gibt einen Zeiger auf den letzten für den Report definierten Style zurück. In Verbindung mit **report4stylePrev** dient die Funktion dazu, die Styles eines Reports rückwärts zu durchlaufen.	
Parameter	report	Ein Zeiger auf den Report, der die Styles enthält.
Rückgaben	**report4styleLast** gibt einen Zeiger auf den letzten definierten Style zurück. Enthält der Report lediglich einen Style, zeigt diese Funktion gleichzeitig auf den ersten Style.	
Siehe auch	**report4stylePrev, report4styleFirst**	

report4styleNext

Aufruf	STYLE4 *report4styleNext(REPORT4 *report)	
Beschreibung	Diese Funktion gibt einen Zeiger auf den nächsten für den Report definierten Style zurück. In Verbindung mit **report4styleFirst** dient die Funktion dazu, die Styles eines Reports vorwärts zu durchlaufen.	
Parameter	report	Ein Zeiger auf den Report, der die Styles enthält.
Rückgaben	Nicht-Null	Ein **STYLE4**-Zeiger auf den nächsten Style des Reports wird zurückgegeben.
	0	Der Report enthält keine weiteren Styles.
Siehe auch	**report4styleFirst, report4stylePrev**	

report4styleSelect

Aufruf int report4styleSelect(REPORT4 *report, STYLE4 *style)

Beschreibung Diese Funktion macht den angegebenen Style zum „aktuellen“ Style. Alle neuen Ausgabeobjekte werden mit diesem Style aufgebaut.

Parameter

report	Ein Zeiger auf den Report, in dem der Style ausgewählt wird.
style	Ein **STYLE4**-Zeiger auf einen bereits vorhandenen Style, der als aktueller Style eingerichtet wird.

Rückgaben

0	Der Style wurde erfolgreich eingestellt.
< 0	Fehler. *report* oder *style* sind ungültig.

report4styleSelected

Aufruf STYLE4 *report4styleSelected(REPORT4 *report)

Beschreibung Diese Funktion gibt einen Zeiger auf den „aktuellen“ Style des Reports zurück.

Parameter

report	Ein Zeiger auf den Report, in dem der Style ausgewählt wird.

Rückgaben **report4styleSelected** gibt einen **STYLE4**-Zeiger auf den vorhandenen und ausgewählten Style zurück. Per Voreinstellung wird immer der zuletzt angelegte Style ausgewählt (es sei denn, Sie verwenden **report4styleSelect**). Haben Sie noch keine Style definiert, gibt die Funktion einen Zeiger auf den voreingestellten Style des Reportmoduls („Normal“) zurück.

report4styleSheetLoad

Aufruf	int report4styleSheetLoad(REPORT4 *report, char *fileName, int overRide)	
Beschreibung	Diese Funktion fügt die Styles eines CodeReporter-Seiten-Layouts in den angegebenen Report ein.	
Parameter	report	Ein Zeiger auf den Report, in den die neuen Styles eingefügt werden.
	fileName	Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Dateinamen (Erweiterung .CRS) des CodeReporter-Seiten-Layouts enthält. Gibt *fileName* kein Laufwerk und/oder Verzeichnis an, wird das aktuelle Verzeichnis durchsucht.
	overRide	Dieser Parameter ist dazu da, Konflikte zwischen gleichnamigen Styles im Report und im Seiten-Layout zu verhindern. Enthält *overRide* einen wahren Wert (nicht-Null), verwendet **report4-styleSheetLoad** bei Benennungskonflikten die Styles des Seiten-Layouts. Enthält der Parameter einen falschen Wert (Null), bleiben die Styles im Report unverändert.
Rückgaben	1	Das Seiten-Layout wurde erfolgreich geladen.
	0	Fehler. Die Datei konnte nicht lokalisiert werden, sie war fehlerhaft oder veraltet.

report4styleSheetSave

Aufruf	int report4styleSheetSave(REPORT4 *report, char *fileName)
Beschreibung	Diese Funktion speichert die Styles des angegebenen Reports als CodeReporter-Seiten-Layout.

Parameter report — Ein Zeiger auf den Report, dessen Styles gespeichert werden sollen.

fileName — Ein null-terminiertes Zeichenarray, das Laufwerk, Verzeichnis und Namen der Datei enthält, in der die Styles abgelegt werden sollen. Geben Sie kein Laufwerk und/oder Verzeichnis an, wird das Seiten-Layout ins aktuelle Verzeichnis geschrieben. Die Erweiterung am Dateinamen lautet .CRS.

Rückgaben 1 — Das Seiten-Layout wurde erfolgreich gespeichert.

0 — Das Seiten-Layout konnte nicht gespeichert werden. Das ist immer dann der Fall, wenn die Datei bereits vorhanden ist.

report4titlePage

Aufruf int report4titlePage(REPORT4 *report, int titlePage)

Beschreibung Mit dieser Funktion erzwingen Sie nach Ausgabe der Titelbereiche des Reports einen Seitenumbruch.

Parameter report — Ein Zeiger auf den Report, für den die Einstellung der Titelseite gelten soll.

titlePage — Dieser Parameter legt fest, ob nach dem Titelbereich ein Seitenumbruch erfolgen soll oder nicht. *titlePage* kann einen der folgenden Werte annehmen:

1 Nach der Ausgabe der Titelbereiche erfolgt ein Seitenumbruch.

0 Nach der Ausgabe der Titelbereiche erfolgt kein Seitenumbruch. Wird diese Funktion nicht aufgerufen, geht das Programm von diesem Wert aus.

Rückgaben	>= 0	Die vorhergehende Einstellung von *titlePage* wird zurückgegeben.
	< 0	Fehler. *report* oder *titlePage* sind ungültig.

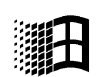
report4titleSummary

Aufruf GROUP4 *report4titleSummary(REPORT4 *report)

Beschreibung Mit dieser Funktion wird ein **GROUP4**-Zeiger auf die Titel-/Schlußgruppe des Reports zurückgegeben.

Parameter	report	Ein Zeiger auf den Report, aus dem die Titel-/Schlußgruppe eingelesen werden soll.

Rückgaben	Nicht-Null	Ein **GROUP4**-Zeiger auf die Titel-/Schlußgruppe des Reports wird zurückgegeben.
	0	Fehler. *report* ist ungültig.



report4toScreen

Aufruf int report4toScreen(REPORT4 *report, int toScreen)

Beschreibung Mit dieser Funktion wird **report4do** angewiesen, ein Fenster zu erzeugen und die Reportausgabe darin vorzunehmen oder statt dessen den Report auf den ausgewählten Drucker zu schicken.

Per Voreinstellung erfolgt die Reportausgabe mit **report4do** in einem Fenster.

Parameter	report	Ein Zeiger auf den Report, der auf dem Bildschirm ausgegeben werden soll.

toScreen — Mit diesem Parameter wird bei Windows-Anwendungen das Ausgabeziel des Reports festgelegt. *toScreen* kann einen der folgenden Werte annehmen:

1 Es wird ein Fenster geöffnet, und der Report wird von der Fensterprozedur gesteuert.

0 Die Reportausgabe erfolgt auf dem ausgewählten Drucker.

Rückgaben

>= 0 — Die vorhergehende Einstellung von *toScreen* wird zurückgegeben.

< 0 — Fehler. *report* oder *toScreen* ist ungültig.

Siehe auch **report4output, report4do**

style4-Funktionen

Die **style4**-Funktionen sind dazu da, eine Reihe von Schriftattributen unter einem gemeinsamen Namen zusammenzufassen; mit **obj4style** lassen sich diese Attribute dann mit Ausgabeobjekten verbinden.

style4color

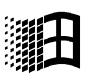
Aufruf int style4color(STYLE4 *style, R4COLORREF color)

Beschreibung Mit dieser Funktion wird die Windows-RGB-Farbe für den angegebenen Style verändert.

Parameter

Parameter	style	Ein Zeiger auf den Style, für den die Farbe eingestellt wird.
	color	Bei diesem Parameter handelt es sich um einen COLORREF-Wert, der die Farbe für den angegebenen Style beschreibt.

Rückgaben	0	Die Farbe wurde erfolgreich eingestellt.
	< 0	Fehler. *style* ist ungültig.

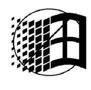


style4create

Aufruf STYLE4 *style4create(REPORT4 *report, R4LOGFONT *font, char *name, R4COLORREF color, int pointSize)

Beschreibung Diese Funktion fügt einen neuen Style mit der angegebenen Windows-Schrift in den Report ein.

Parameter	report	Ein Zeiger auf den Report, mit dem der neue Style verbunden werden soll.
	font	Ein Zeiger auf eine Windows-LOGFONT-Struktur, die die für den neuen Style verwendete Schriftart beschreibt.
	name	Ein null-terminiertes Zeichenarray, das den Namen des neuen Styles enthält. *name* kann auf bis zu 19 Zeichen zeigen.
	color	Bei diesem Parameter handelt es sich um einen COLORREF-Wert, der die Farbe für den neuen Style beschreibt.
	pointSize	Die Größe der für den neuen Style verwendeten Schrift in Punkten.
Rückgaben	Nicht-Null	Der neue Style wurde erfolgreich eingerichtet. Der zurückgegebene Zeiger darf als gültig betrachtet und von anderen Funktionen des Reportmoduls verwendet werden.
	0	Fehler. Der Style wurde nicht angelegt.



style4create

Aufruf STYLE4 *style4create(REPORT4 *report, char *name,
int beforeLen, char *beforeCodes, int afterLen, char *afterCodes)

Beschreibung Diese Funktion erzeugt einen Non-Windows-Style, der die druckerspezifischen Steuerzeichen aufnimmt, die eine bestimmte Druckerschrift beschreiben.

Parameter	report	Ein Zeiger auf den Report, für den der neue Style gelten soll.
	name	Ein null-terminiertes Zeichenarray, das den Namen des neuen Styles enthält. *name* kann auf bis zu 19 Zeichen zeigen.
	beforeLen	Die Länge des Zeichenarrays der Druckersteuerzeichen, die vor dem Text des Ausgabeobjekts übermittelt werden.
	beforeCodes	Ein Zeichenarray, das die Druckersteuerzeichen enthält, die vor dem Text des Ausgabeobjekts übermittelt werden. Diese Steuerzeichen sollten ein bestimmtes Druckattribut oder eine Schriftart aktivieren.
	afterLen	Die Länge des Zeichenarrays der Druckersteuerzeichen, die nach dem Text des Ausgabeobjekts übermittelt werden.
	afterCodes	Ein Zeichenarray, das die Druckersteuerzeichen enthält, die nach dem Text des Ausgabeobjekts übermittelt werden. Diese Steuerzeichen sollten ein bestimmtes Druckattribut oder eine Schriftart deaktivieren.
Rückgaben	Nicht-Null	Der neue Style wurde erfolgreich eingerichtet. Der zurückgegebene Zeiger darf als gültig betrachtet und von anderen Funktionen des Reportmoduls verwendet werden.
	0	Fehler. Der Style wurde nicht angelegt.

style4delete

Aufruf int style4delete(REPORT4 *report, char *styleName)

Beschreibung Diese Funktion löscht den bezeichneten Style aus dem Report und gibt den mit dem Style verbundenen Speicher frei. Handelte

es sich bei dem gelöschten Style um den aktuellen, wird der erste Style des Reports nun zum aktuellen Style.

Parameter	report	Ein Zeiger auf den Report, der den zu löschenden Style enthält.
	styleName	Ein null-terminiertes Zeichenarray, das den Namen des zu löschenden Styles enthält. **style4delete** durchläuft die Styles des Reports und vergleicht deren Namen mit *styleName*. Kommt es zu einer Übereinstimmung, wird der Style aus dem Report entfernt.
Rückgaben	1	Style erfolgreich lokalisiert und gelöscht.
	0	Ein Style mit Namen *styleName* wurde im Report nicht gefunden.

style4free

Aufruf int style4free(REPORT4 *report, STYLE4 *style)

Beschreibung Diese Funktion löscht den angegebenen Style aus dem Report und gibt den damit verbundenen Speicher frei. Handelte es sich bei dem gelöschten Style um den aktuellen, wird der erste Style des Reports zum aktuellen Style.

Parameter	report	Ein Zeiger auf den Report, der den zu löschenden Style enthält.
	style	Ein Zeiger auf den zu löschenden Style.
Rückgaben	1	Der Style wurde erfolgreich gelöscht.
	0	Fehler. *report* und/oder *style* sind ungültig.

style4index

Aufruf	STYLE4 *style4index(REPORT4 *report, int styleIndex)	
Beschreibung	Diese Funktion gibt einen **STYLE4**-Zeiger auf den Style in der *styleIndex*ten Position zurück.	
Parameter	report	Ein Zeiger auf den Report, der den gewünschten Style enthält.
	styleIndex	Ein Index des internen Seiten-Layouts des Reports. Mit dieser Funktion wird rasch ein Zeiger auf den *styleIndex*ten Style des Reports abgerufen. Dabei ist der erste Style Style 1 (eins). Der Zeiger wird von den **report4pageObj**-Funktionen benutzt.
Rückgaben	Nicht-Null	Ein **STYLE4**-Zeiger auf den angegebenen Style.
	0	*styleIndex* ist größer als die Anzahl der im Report vorhandenen Styles oder null.
Siehe auch	**style4lookup, report4pageObjFirst**	

style4lookup

Aufruf	STYLE4 *style4lookup(REPORT4 *report, char *styleName)	
Beschreibung	Diese Funktion gibt einen **STYLE4**-Zeiger auf den Style mit dem angegebenen Namen zurück.	
Parameter	report	Ein Zeiger auf den Report, der den gewünschten Style enthält.
	styleName	Ein null-terminiertes Zeichenarray, das den Namen des Styles enthält, der gesucht wird.

Rückgaben	Nicht-Null	Ein **STYLE4**-Zeiger auf den angegebenen Style.
	0	Ein Style namens *styleName* konnte im angegebenen Report nicht lokalisiert werden.

Siehe auch **style4index**

total4-Funktionen

Mit den **total4**-Funktionen werden die für eine Summe erforderlichen Informationen angegeben. Ist er erst vorhanden, können Sie den **TOTAL4**-Zeiger zusammen mit **obj4totalCreate** einsetzen, um ein Summenausgabeobjekt zum Report hinzuzufügen.

total4addCondition

Aufruf int total4addCondition(TOTAL4 *total, char *addConditionSrc, int logical)

Beschreibung Mit dieser Funktion wird eine bedingte Summierung des Summenausgabeobjekts angegeben.

Handelt es sich bei *addConditionSrc* um einen logischen dBASE-Ausdruck (und *logical* ist nicht-Null), wird die Summe immer dann gebildet, wenn die Bedingung auf einen wahren Wert hinausläuft.

Handelt es sich bei *addConditionSrc* um einen anderen Ausdruckstyp (und *logical* ist null), wird die Summe gebildet, sobald sich der Wert der ausgewerteten Bedingung ändert.

Wird diese Funktion nicht aufgerufen, wird die Summe für jeden Datensatz der zusammengesetzten Datendatei gebildet.

Parameter

total — Ein Zeiger auf die Summe, für die die bedingte Summierung gelten soll.

addConditionSrc — Ein null-terminiertes Zeichenarray, das einen dBASE-Ausdruck enthält, mit dem der Zeitpunkt der Summenbildung festgelegt wird. Bei diesem Ausdruck kann es sich, bis auf Memo, um jeden beliebigen Typ handeln. Hat der Ausdruck als Ergebnis einen wahren Wert (und *logical* ist nicht-

Null) oder ändert sich der ausgewertete Ausdruck (und *logical* ist null), wird die Summe gebildet.

logical — Mit diesem Flag wird bestimmt, ob die Summe bei einer logischen Bedingung oder bei einer Wertänderung gebildet werden soll. Enthält *logical* einen wahren Wert (nicht-Null), wird davon ausgegangen, daß das Ergebnis von *addConditionSrc* ein logischer Wert ist. Wenn *logical* einen falschen Wert (null) enthält, wird die Summe lediglich bei einer Änderung des ausgewerteten Ausdrucks gebildet.

addConditionSrc kann als Ergebnis einen logischen Wert haben, und gleichzeitig kann *logical* falsch (null) sein. In diesem Fall wird die Summe gebildet, sobald die Auswertung des Ausdrucks sich von `.TRUE.` nach `.FALSE.` und von `.FALSE.` nach `.TRUE.` verändert.

Rückgaben

0 — Die Bedingung wurde erfolgreich zum Summenausgabeobjekt hinzugefügt.

< 0 — Fehler. Ein ungültiger Parameter wurde übergeben; *addConditionSrc* hatte als Ergebnis keinen logischen Wert, als *logical* auf wahr (nicht-Null) eingestellt war, bzw. *addConditionSrc* konnte nicht ausgewertet werden.

Siehe auch **obj4totalCreate**, die CodeBase-5-Funktion **expr4calc_create**

total4create

Aufruf

```
TOTAL4 *total4create( REPORT4 *report, char *totalName,
                      char *totalExpr, int type, char *resetExpr )
```

Beschreibung Diese Funktion definiert eine Summe, die in **obj4totalCreate** verwendet wird.

Parameter

report	Ein Zeiger auf den Report, zu dem die Summe hinzugefügt werden soll.
totalName	Ein null-terminiertes Zeichenarray, das einen sprechenden Namen enthält, mit dem in anderen dBASE-Ausdrücken auf die Summe verwiesen wird. Diese Bezeichnung darf keine Leerstellen enthalten.
totalExpr	Ein null-terminiertes Zeichenarray, das einen numerischen dBASE-Ausdruck enthält, aufgrund dessen die Summe erzeugt wird. Dabei kann es sich einfach um ein Feld der Datendatei oder eine Berechnung handeln, die mit der CodeBase-5-Funktion **expr4calc_create** erstellt worden ist.

Hinweis: Alle Felder der Datendatei, auf die *totalExpr* verweist, müssen einen Feldkennzeichner haben.

type — Mit diesem Flag wird festgelegt, wie die Summe bei der Auswertung des Summenausgabeobjekts ihren Wert behält. *type* kann einen der folgenden Konstantenwerte annehmen:

total4average	Diese Konstante erzeugt eine Summe, die den arithmetischen Mittelwert (Durchschnitt) des Ausdrucks *totalExpr* bildet.
total4highest	Diese Konstante erzeugt eine Summe, die den höchsten für *totalExpr* ermittelten Wert speichert.
total4lowest	Diese Konstante erzeugt eine Summe, die den niedrigsten Wert für *totalExpr* speichert.
total4sum	Diese Konstante erzeugt eine Summe, die eine arithmetische Summe aller für den Ausdruck *totalExpr* ermittelten Werte bildet.

	resetExpr	Ein null-terminiertes Zeichenarray, das einen dBASE-Ausdruck enthält, mit dem festgelegt wird, wann das Summenausgabeobjekt auf seinen Ausgangswert zurückgesetzt wird.
Rückgaben	Nicht-Null	Ein Zeiger auf eine erfolgreich erzeugte Summe wird zurückgegeben.
	0	Beim Analysieren des dBASE-Ausdrucks sind Schwierigkeiten aufgetreten.

Siehe auch **obj4totalCreate**, die CodeBase-5-Funktion **expr4calc_create**

total4free

Aufruf void total4free(TOTAL4 *total)

Beschreibung Diese Low-Level-Funktion gibt den gesamten mit der Definition der Summe verbundenen Speicher frei. Sie wird automatisch von **obj4totalFree** aufgerufen.

Parameter total — Ein Zeiger auf die Summendefinition, die erzeugt werden soll.

Siehe auch **obj4totalFree, total4create**

Anhang A: dBASE-Funktionen

dBASE-Ausdrucksfunktionen

Die unten aufgeführten Funktionen können als dBASE-Ausdruck oder als Teil eines dBASE-Ausdrucks benutzt werden. Wie Operatoren, Konstanten und Felder geben auch Funktionen einen Wert zurück. Die Funktionen haben stets einen Namen, dem eine linke und eine rechte Klammer folgt. In den Klammern können Werte (Parameter) stehen.

Liste der Funktionen

CTOD(Zeichen_Wert)

Die Funktion „character to date“ konvertiert einen Zeichenwert in einen Datumswert.

Beispiel: "CTOD("11/30/88")"

Die Zeichendarstellung erfolgt immer im von **Code4::dateFormat** angegebenen Format, das auf „MM/DD/YY“ voreingestellt ist.

DATE()

Gibt das Systemdatum zurück.

DAY(Datums_Wert)

Gibt den Tag des Datumsparameters als numerischen Wert von „1“ bis „31“ zurück.

Beispiel: "DAY(DATE())"

Gibt „30“ zurück, wenn es sich um den 30. Tag des Monats handelt.

DEL()

Gibt „*“ zurück, wenn der aktuelle Datensatz zum Löschen markiert ist. Sonst wird „ “ zurückgegeben.

DELETED()

Gibt .TRUE. zurück, wenn der aktuelle Datensatz zum Löschen markiert ist.

DESCEND()

(Nur bei Clipper-Kompatibilität) Gibt die Komplementversion eines Ausdrucks zurück.

DTOC(Datums_Wert)

Die Funktion „date to character“ konvertiert einen Datumswert in einen Zeichenwert. Das Format des resultierenden Zeichenwertes wird von **Code4::dateFormat** angegeben, das auf „MM/DD/YY“ voreingestellt ist.

Beispiel: "DTOC(DATE())"

Gibt den Zeichenwert „04/08/93“ zurück, wenn das Datum der 8. April 1993 ist.

DTOS(Datums_Wert)

Die Funktion „date to string“ konvertiert einen Datumswert in einen Zeichenwert. Das Format des resultierenden Zeichenwertes ist „CCYYMMDD“.

Beispiel: "DTOS(DATE())"

Gibt den Zeichenwert „19870530“ zurück, wenn das Datum der 30. Mai 1987 ist.

IIF(Log_Wert, Wahr_Result, Falsch_Result)

Ist „Log_Wert“ .TRUE., gibt IIF den Wert „Wahr_Result“ zurück; sonst gibt IIF den Wert „Falsch_Result“ zurück. Sowohl „Wahr_Result“ als auch „Falsch_Result“ müssen dieselbe Länge haben und vom selben Typ sein; sonst wird ein Fehler generiert.

Beispiel 1: "IIF(WERT < 0, "Kleiner als Null", "Größer als Null ")"
Beispiel 2: "IIF(NAME = "Hans", "Name ist Hans","Nicht Hans ")"

LTRIM(Zeichen_Wert)

Diese Funktion entfernt die führenden Leerzeichen aus dem Ausdruck.

MONTH(Datums_Wert)

Gibt den Monat des Datumsparameters als numerischen Wert zurück.

Beispiel: "MONTH(DT_FELD)"

Gibt „12“ zurück, wenn der Monat des Datumsfeldes Dezember ist.

PAGENO()

Wird das Reportmodul bzw. CodeReporter eingesetzt, gibt diese Funktion die aktuelle Seitenzahl des Reports zurück.

RECCOUNT()

Die Funktion „record count“ gibt die Gesamtzahl der Datensätze in der Datenbank zurück.

Beispiel: "RECCOUNT()"

Gibt „10“ zurück, wenn die Datenbank 10 Datensätze enthält.

RECNO()

Die Funktion „record number“ gibt die Datensatznummer des aktuellen Datensatzes zurück.

STOD(Zeichen_Wert)

Die Funktion „string to date“ konvertiert einen Zeichenwert in einen Datumswert.

Beispiel: "STOD("19881130")"

Die Zeichendarstellung erfolgt im Format „CCYYMMDD“.

STR(Zahl, Länge, Nachkomma)

Die „string“-Funktion konvertiert einen numerischen Wert in einen Zeichenwert. „Länge“ ist die Anzahl der Zeichen in der neuen Zeichenkette, das Dezimalkomma inbegriffen. „Nachkomma“ ist die Anzahl der gewünschten Nachkommastellen. Ist die Zahl für den zugewiesenen Platz zu groß, werden Sternchen (*) zurückgegeben.

Beispiel: "STR(5.7, 4, 2)" gibt "'5.70'" zurück.

Die Zahl 5.7 wird in eine Zeichenkette konvertiert, die aus vier Zeichen besteht. Außerdem sind zwei Nachkommastellen vorhanden.

Beispiel: "STR(5.7, 3, 2)" gibt "'***'" zurück.

Die Zahl 5.7 paßt nicht in eine Zeichenkette, die aus drei Zeichen besteht, wenn sie zwei Nachkommastellen haben soll. Infolgedessen werden Sternchen eingesetzt.

SUBSTR(Zeichen_Wert, Start_Position, Anzahl_Zeichen)

Ein Teilbereich des Zeichenwerts wird zurückgegeben. Der Teilbereich wird „Anzahl_Zeichen“ lang sein und auf dem Zeichen „Start_Position“ von „Zeichen_Wert“ beginnen.

Beispiel 1: "SUBSTR("ABCDE", 2, 3)" gibt "'BCD'" zurück.
Beispiel 2: "SUBSTR("Hr. Greser", 5, 1) gibt "'G'" zurück.

TIME()

Die Zeitfunktion gibt die Systemzeit in Zeichendarstellung zurück. Sie benutzt das Format „HH:MM:SS“.

Beispiel 1: "TIME()" gibt "12:00:00" zurück, wenn es Mittag ist.
Beispiel 2: "TIME()" gibt "13:30:00" zurück, wenn es 13.30 Uhr nachmittags ist.

TRIM()

Diese Funktion entfernt alle Leerzeichen am Ende eines Ausdrucks.

UPPER(Zeichen_Wert)

Eine Zeichenkette wird auf Großschreibung konvertiert, und das Ergebnis wird zurückgegeben.

VAL(Zeichen_Wert)

Die Funktion „value“ konvertiert einen Zeichenwert in einen numerischen Wert.

Beispiel 1: "VAL('10')" gibt "10" zurück.
Beispiel 2: "VAL("-8.7")" gibt "-8.7" zurück.

YEAR(Datums_Wert)

Gibt das Jahr des Datumsparameters als numerischen Wert zurück.

Beispiel: "YEAR(STOD('19920830'))" gibt "1992" zurück.

Anhang B: Tastenkombinationen

Bei CodeReporter ist eine Microsoft-kompatible Maus erforderlich. Bestimmte Operationen, wie z. B. das Plazieren von Ausgabeobjekten, können ausschließlich mit einer Maus vorgenommen werden.

Die meisten anderen Operationen lassen sich sowohl per Maus als auch über die Tastatur ausführen. Der vorliegende Anhang enthält eine systematische Übersicht über die Tastenfolgen und die entsprechenden Befehle.

Menübeschleunigungstasten

Menübeschleunigungstasten sind Tasten, mit denen Menüoperationen schnell ausgeführt werden können. Das verkürzt die Reaktionszeit bei der Ausführung zahlreicher CodeReporter-Funktionen.

Strg-A	**BEREICH | NEUER KOPFBEREICH**. Diese Tastenfolge legt einen neuen Kopfbereich für die ausgewählte Gruppe an.
Strg-C **Strg-Einfg**	**BEARBEITEN | KOPIEREN**. Eine Kopie des gerade ausgewählten Ausgabeobjekts wird in der Windows-Zwischenablage angelegt, aus der sie aus CodeReporter mit **BEARBEITEN | EINFÜGEN** wieder abgerufen werden kann.
Strg-E	**AUSRICHTEN | ZENTRIERT**. Diese Tastenfolge richtet das ausgewählte Objekt so aus, daß es mittig im Report steht. Sind mehrere Objekte ausgewählt, werden alle ausgewählten Ausgabeobjekte in bezug auf das erste mittig ausgerichtet.
Strg-F	**BEREICH | NEUER FUSSBEREICH**. Diese Tastenfolge legt einen neuen Fußbereich für die ausgewählte Gruppe an.
Strg-G	**GRUPPE | NEU**. Diese Tastenfolge ruft die Dialogbox „Gruppendefinition“ auf und legt eine neue Gruppe mit Kopf- und Fußbereich an.
Strg-H	**FEINEINSTELLUNG | WAAGERECHT AUSTREIBEN**. Diese Tastenfolge versetzt alle ausgewählten Ausgabeobjekte so, daß die

Horizontalabstände zwischen ihnen gleich groß sind. Dabei dienen das erste und das letzte ausgewählte Ausgabeobjekt als Fixpunkte.

Strg-L AUSRICHTEN | LINKSBÜNDIG. Diese Tastenfolge richtet alle ausgewählten Ausgabeobjekte so aus, daß ihre linke Seite eine Flucht mit der linken Seite des ersten ausgewählten Objekts bildet.

Strg-M GRUPPE | BEARBEITEN. Diese Tastenfolge ruft die Dialogbox „Gruppendefinition“ für die zur Zeit ausgewählte Gruppe auf.

Strg-O BEREICH | BEREICH BEARBEITEN. Diese Tastenfolge ruft die Dialogbox „Bereich bearbeiten“ für den ausgewählten Bereich auf.

Strg-P DATEI | DRUCKEN. Diese Tastenfolge ruft die Dialogbox „Drucken“ auf, über die der aktuelle Report sich auf dem angegebenen Drucker ausgeben läßt.

Strg-R AUSRICHTEN | RECHTSBÜNDIG. Diese Tastenfolge richtet alle ausgewählten Ausgabeobjekte so aus, daß ihre rechte Seite eine Flucht mit der rechten Seite des ersten ausgewählten Objekts bildet.

Strg-S DATEI | SPEICHERN. Diese Tastenfolge legt den aktuellen Report auf der Platte ab. Wurde der Report bislang noch nicht gesichert, bittet CodeReporter um Eingabe eines Dateinamens.

Strg-T FEINEINSTELLUNG | SENKRECHT AUSTREIBEN. Diese Tastenfolge versetzt alle ausgewählten Ausgabeobjekte so, daß die Vertikalabstände zwischen ihnen gleich groß sind. Dabei dienen das erste und das letzte ausgewählte Ausgabeobjekt als Fixpunkte.

Strg-V
Groß-Einfg BEARBEITEN | EINFÜGEN. CodeReporter wird für Ausgabeobjekte in der Windows-Zwischenablage in den Einfügemodus versetzt. Enthält die Zwischenablage einen Text aus einer anderen Anwendung, erzeugt CodeReporter ein Textausgabeobjekt. Enthält die Zwischenablage eine Bitmap, wird diese als statisches Grafikobjekt in den Report eingefügt.

Strg-W	**DATEI | DRUCKBILD EINSEHEN**. Diese Tastenfolge öffnet ein ganzseitiges Bildschirmfenster und gibt den Report darin aus.
Strg-X **Groß-Entf**	**BEARBEITEN | AUSSCHNEIDEN**. Die zur Zeit ausgewählten Ausgabeobjekte werden aus dem Report gelöscht und in der Windows-Zwischenablage abgelegt, aus der sie mit **BEARBEITEN | EINFÜGEN** von CodeReporter abgerufen werden können.
Entf	**OBJEKT | LÖSCHEN**. Diese Taste löscht das zur Zeit ausgewählte Ausgabeobjekt. Ist kein Objekt ausgewählt, geschieht nichts.
Esc	**OBJEKT | KEIN**. Beim Betätigen dieser Taste verläßt CodeReporter den Einfügemodus.

Report-Entwicklungsbildschirm

Im Report-Entwicklungsbildschirm können Sie statt der Maus auch folgende Tasten betätigen, um dieselben Operationen auszuführen.

Tab	Die Tab-Taste markiert das nächste Objekt eines Bereichs als „ausgewähltes“ Ausgabeobjekt. Betätigen Sie die Tab-Taste, wenn die Markierung auf dem letzten Objekt eines Bereichs steht, wird das erste Objekt ausgewählt. Ist bislang noch kein Objekt ausgewählt, wird beim Betätigen der Tab-Taste das erste Objekt des Bereichs ausgewählt.
Groß-Tab	Die Tastenkombination Groß-Tab bewirkt dasselbe wie die Tab-Taste, mit dem Unterschied allerdings, daß die Markierung den ausgewählten Bereich rückwärts durchläuft.
Strg-Tab	Mit der Tastenkombination Strg-Tab werden im ausgewählten Bereich mehrere Ausgabeobjekte markiert. Dabei wird die Markierung auf das nächste Objekt des Bereichs gesetzt, während alle bereits markierten Objekte weiterhin ausgewählt bleiben.
Groß-Strg-Tab	Die Tastenkombination Groß-Strg-Tab bewirkt dasselbe wie die Kombination Strg-Tab, mit dem Unterschied allerdings, daß die Markierung den ausgewählten Bereich rückwärts durchläuft.

Bild8
Groß-8
Mit diesen Tasten blättern Sie im Report-Entwicklungsbildschirm eine Seite vor. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tasten nichts.

Bild9
Groß-9
Mit diesen Tasten blättern Sie im Report-Entwicklungsbildschirm eine Seite zurück. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tasten nichts.

Groß-7
Diese Tastenkombination verschiebt den Report-Entwicklungsbildschirm so, daß der linke Rand des Fensters zu sehen ist. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tastenfolge nichts.

Groß-6
Diese Tastenkombination verschiebt den Report-Entwicklungsbildschirm so, daß der rechte Rand des Fensters zu sehen ist. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tastenfolge nichts.

Strg-7
Diese Tastenkombination verschiebt den Report-Entwicklungsbildschirm ein klein wenig nach links. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tastenfolge nichts.

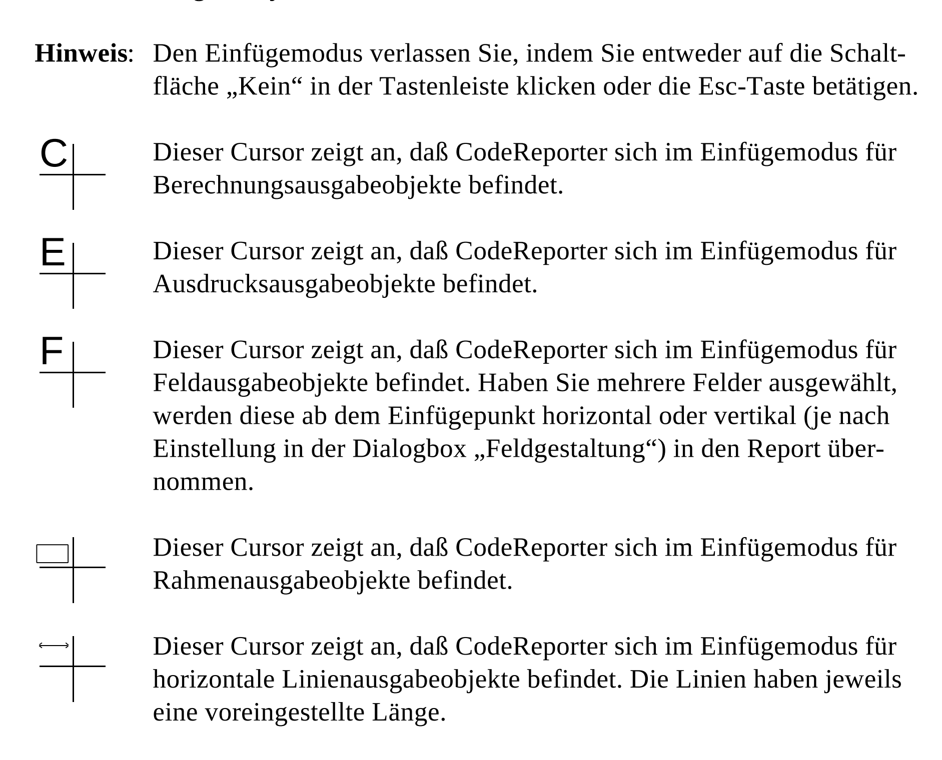
Strg-6
Diese Tastenkombination verschiebt den Report-Entwicklungsbildschirm ein klein wenig nach rechts. Sind alle Elemente des Reports auf dem aktuellen Bildschirm zu sehen, geschieht beim Betätigen dieser Tastenfolge nichts.

Return
Bei Betätigen dieser Taste wird das Objektmenü für das aktuelle Ausgabeobjekt aufgerufen. Sind keine Objekte ausgewählt, geschieht beim Betätigen dieser Taste nichts.

Anhang C: Cursorformen

Wird CodeReporter bei einem bestimmten Typ von Ausgabeobjekt in den Einfügemodus versetzt, ändert sich der Mauszeiger. Er zeigt dann sowohl den Einfügemodus als auch den Typ des Objekts an, das hinzugefügt werden soll.

Im folgenden führen wir die verschiedenen Cursorformen und die Objekte auf, bei denen sie auftreten. Jeder Cursor hat ein „Fadenkreuz“. Der Schnittpunkt der beiden Linien bestimmt nach dem Einfügen die Position der oberen linken Ecke des neuen Ausgabeobjekts.



Hinweis: Den Einfügemodus verlassen Sie, indem Sie entweder auf die Schaltfläche „Kein“ in der Tastenleiste klicken oder die Esc-Taste betätigen.

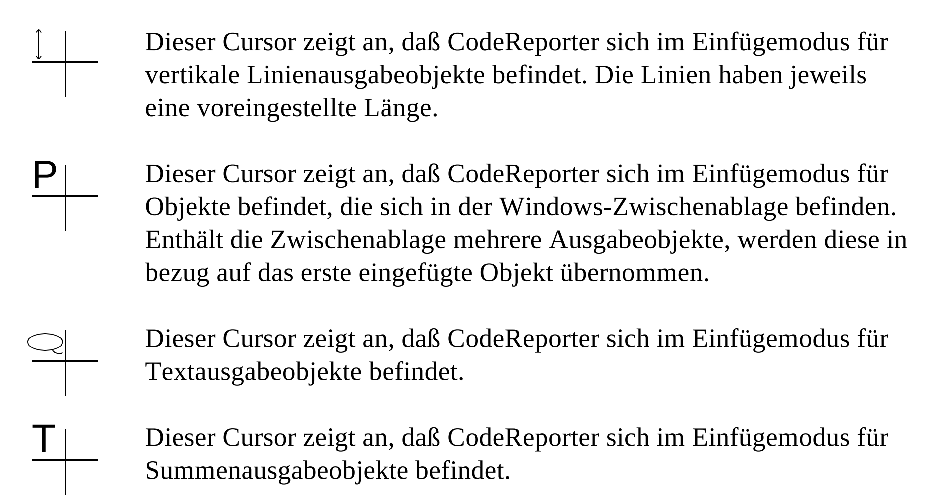
C — Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Berechnungsausgabeobjekte befindet.

E — Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Ausdrucksausgabeobjekte befindet.

F — Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Feldausgabeobjekte befindet. Haben Sie mehrere Felder ausgewählt, werden diese ab dem Einfügepunkt horizontal oder vertikal (je nach Einstellung in der Dialogbox „Feldgestaltung“) in den Report übernommen.

Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Rahmenausgabeobjekte befindet.

Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für horizontale Linienausgabeobjekte befindet. Die Linien haben jeweils eine voreingestellte Länge.



Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für vertikale Linienausgabeobjekte befindet. Die Linien haben jeweils eine voreingestellte Länge.

Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Objekte befindet, die sich in der Windows-Zwischenablage befinden. Enthält die Zwischenablage mehrere Ausgabeobjekte, werden diese in bezug auf das erste eingefügte Objekt übernommen.

Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Textausgabeobjekte befindet.

Dieser Cursor zeigt an, daß CodeReporter sich im Einfügemodus für Summenausgabeobjekte befindet.

Anhang D: ASCII-Tabelle (nicht vollständig)

Im folgenden führen wir die verbreitetsten Druckersteuerzeichen mit der entsprechenden Hexadezimalnotierung auf. Da die Hexadezimalwerte in den meisten Druckerhandbüchern aufgeführt werden, mag die Übersicht als zusätzliche Referenz zum raschen Nachschlagen dienen.

ASCII	Dez	Hex	ASCII	Dez	Hex	ASCII	Dez	Hex
ESC	27	1B	:	58	3A	T	84	54
!	33	21	;	59	3B	U	85	55
"	34	22	<	60	3C	V	86	56
#	35	23	=	61	3D	W	87	57
$	36	24	>	62	3E	X	88	58
%	37	25	?	63	3F	Y	89	59
&	38	26	@	64	40	Z	90	5A
'	39	27	A	65	41	[	91	5B
(	40	28	B	66	42	\	92	5C
)	41	29	C	67	43	]	93	5D
*	42	2A	D	68	44	^	94	5E
+	43	2B	E	69	45	_	95	5F
,	44	2C	F	70	46	`	96	60
-	45	2D	G	71	47	a	97	61
.	46	2E	H	72	48	b	98	62
/	47	2F	I	73	49	c	99	63
0	48	30	J	74	4A	d	100	64
1	49	31	K	75	4B	e	101	65
2	50	32	L	76	4C	f	102	66
3	51	33	M	77	4D	g	103	67
4	52	34	N	78	4E	h	104	68
5	53	35	O	79	4F	i	105	69
6	54	36	P	80	50	j	106	6A
7	55	37	Q	81	51	k	107	6B
8	56	38	R	82	52	l	108	6C
9	57	39	S	83	53	m	109	6D

<table>
<tr><th>ASCII</th><th>Dez</th><th>Hex</th><th>ASCII</th><th>Dez</th><th>Hex</th><th>ASCII</th><th>Dez</th><th>Hex</th></tr>
<tr><td>n</td><td>110</td><td>6E</td><td>t</td><td>116</td><td>74</td><td>z</td><td>122</td><td>7A</td></tr>
<tr><td>o</td><td>111</td><td>6F</td><td>u</td><td>117</td><td>75</td><td>{</td><td>123</td><td>7B</td></tr>
<tr><td>p</td><td>112</td><td>70</td><td>v</td><td>118</td><td>76</td><td>|</td><td>124</td><td>7C</td></tr>
<tr><td>q</td><td>113</td><td>71</td><td>w</td><td>119</td><td>77</td><td>}</td><td>125</td><td>7D</td></tr>
<tr><td>r</td><td>114</td><td>72</td><td>x</td><td>120</td><td>78</td><td>~</td><td>126</td><td>7E</td></tr>
<tr><td>s</td><td>115</td><td>73</td><td>y</td><td>121</td><td>79</td><td></td><td></td><td></td></tr>
</table>

Anhang E: Fehlercodes

Diesem Teil des Anhangs können Sie die Fehlercodes entnehmen, die die Code-Reporter-API-Funktionen beim Auftreten eines Fehlers zurückgeben. Die übrigen Fehlercodes schlagen Sie bitte im CodeBase-Referenzhandbuch nach.

Name der Konstanten	Wert	Bedeutung
e4report	-810	**Reportfehler** Allgemeiner Fehler beim Report. Alle Reportfehler, die hier nicht gesondert aufgeführt werden, sind **e4report**-Fehler.
e4style_create	-811	**Fehler beim Anlegen des Styles** Der Style konnte aufgrund eines bereits vorhandenen Stylenamens oder weil der Speicher nicht ausreichte, nicht angelegt werden.
e4style_select	-812	**Fehler beim Auswählen des Styles** Der **STYLE4**-Zeiger auf eine Style-Auswahlfunktion ist ungültig.
e4style_index	-813	**Fehler beim Lokalisieren des Styles** Der angegebene Style ist nicht auffindbar. Entweder wurde ein ungültiger **STYLE4**-Zeiger oder ein nicht vorhandener Stylename an eine Style-Suchfunktion übergeben.
e4area_create	-814	**Fehler beim Anlegen des Bereichs** Der Bereich konnte nicht angelegt werden, weil ihm entweder kein Speicher zugewiesen oder der Unterdrückungsausdruck nicht analysiert werden konnte.
e4group_create	-815	**Fehler beim Anlegen der Gruppe** Die Gruppe konnte aufgrund von zu wenig Speicher oder eines ungültigen **GROUP4**-Zeigers nicht angelegt werden.

e4group_expr	-816	**Fehler beim Einrichten des Gruppenrücksetzausdrucks** Der Gruppenrücksetzausdruck konnte nicht richtig analysiert werden.
e4total_create	-817	**Fehler beim Erzeugen der Summe** Das Summenausgabeobjekt konnte aufgrund von Speichermangel, aufgrund einer ungültigen numerischen Berechnung oder aufgrund eines ungültigen Rücksetzausdrucks nicht erzeugt werden.
e4obj_create	-818	**Fehler beim Erzeugen des Objekts** Das Ausgabeobjekt konnte aufgrund von Speichermangel oder eines ungültigen **AREA4**-Zeigers nicht erzeugt werden.
e4rep_win	-819	**Fehler bei der Ausgabe unter Windows** Ein Fehler hat sich bei der Anmeldung der Fensterklasse zur Ausgabe ereignet, oder „CreateWindow“ ist fehlgeschlagen.
e4rep_out	-820	**Fehler bei der Reportausgabe** Die Auswertung eines Ausgabeobjekts hat einen Fehler verursacht. Die Ursache dafür ist im allgemeinen zu wenig Speicher.
e4rep_save	-821	**Fehler beim Speichern des Reports** Beim Abspeichern einer Reportdatei hat sich ein Fehler ereignet. Die Ursache dafür können eine bereits vorhandene Datei, fehlender Plattenplatz oder Schwierigkeiten beim Anlegen der Datei sein.
e4rep_ret	-822	**Fehler beim Einlesen des Reports** Die Reportdatei konnte nicht von der Platte eingelesen werden. Das kann daran liegen, daß die Top-Master-Datendatei nicht auffindbar oder die Reportdatei zerstört worden ist.

e4rep_data	-823	**Fehler beim Anlegen der Ausgabedatendatei** Der Report konnte nicht in die Ausgabedaten-datei übertragen werden. Ursache dafür können ein Fehler beim Anlegen der Datei oder ein Fehler beim Ablegen der Daten sein.

Anhang F: CodeBasic-API

Der vorliegende Abschnitt erläutert die unter CodeBasic verfügbaren Funktionen, mit denen Reports ausgeführt und modifiziert werden können. Mit diesen Funktionen können Sie u. a. Reports von der Platte einlesen, sie sofort anzeigen lassen und bearbeiten und die Änderungen auf die Platte zurückschreiben.

Bis auf drei Funktionen handelt es sich um **report4**-Funktionen. Die drei **relate4**-Funktionen lassen sich in Verbindung mit der Menüoption **DATEI | RELATION SPEICHERN** einsetzen, um komplexe Relationen aufzubauen, die auf der Platte abgelegt werden können. Diese Relationen können dann später für die Verwendung in einer CodeBasic-Anwendung eingelesen werden.

Das folgende Programm REPTEST.BAS testet eine BASIC-Anwendung.

```
Sub ReportTest (cb As Long, Report As Form)

   Dim lReport As Long

   'Reportdatei TUT1.REP einlesen
   lReport = report4retrieve(cb, App.Path + "TUT1", 1, App.Path)

   'Auf Fehler überprüfen
   If lReport = 0 Then
      rc = code4errorCode(cb, 0)
      Exit Sub
   End If

   'Einige Einstellungen des Reports verändern
   rc = report4caption(lReport, "Neuer Titel")
   rc = report4currency(lReport, "DM")

   'Relationsset des Reports verändern
   rc = report4querySet(lReport, "STUDENT->L_NAME > 'R'")
   rc = report4sortSet(lReport, "STUDENT->L_NAME")

   'Änderungen in einer neuen Datei ablegen
   rc = report4save(lReport, App.Path + "\STUDENT2", 1)

   'Eltern-Handle des Reports vor der Anzeige einstellen
   rc = report4parent(lReport, Report.hWnd)
```

```
    'Report auf dem Bildschirm ausgeben
    rc = report4do(lReport)

    'Drucker zur Ausgabe einstellen
    Call report4printerSelect(lReport)

    'Druckerausgabe
    rc = report4toScreen(lReport, 0)
    rc = report4do(lReport)

    'Speicher freigeben und Dateien schließen
    Call report4free(lReport, 1, 1)

End Sub
```

relate4retrieve

Aufruf RELATE4& = relate4retrieve(CODE4&, fileName$, openFiles%, dataPathName$)

Beschreibung Diese Funktion liest eine Relationsdatei ein und baut die von CodeReporter erzeugte oder mit **relate4save** gespeicherte Relation auf. Beim Laden der Relationsdatei mit **relate4retrieve** kann man gleichzeitig die entsprechenden Datendateien öffnen.

Hinweis: Bei komplizierten Relationen sollten Sie CodeReporter mit der Menüoption **DATEI | RELATION SPEICHERN** einsetzen, um die Relation aufzubauen und auf der Platte abzulegen. Diese Relation können Sie dann später mit **relate4retrieve** in Ihre Anwendung einlesen.

Parameter

CODE4	Ein Zeiger auf die **CODE4**-Struktur der Anwendung; er dient der Speicherverwaltung und Fehlerbehandlung.
fileName	Eine Zeichenkette, die den Namen (einschließlich Laufwerk und Verzeichnis) der Relationsdatei enthält. Da stets die Erweiterung .REL verwendet wird, brauchen Sie keinen Dateityp einzugeben.
openFiles	Hat *openFiles* einen wahren Wert (1), versucht **relate4retrieve**, die in der abgespeicherten Relationsdatei genannten Daten-, Index- und Memodateien zu öffnen, falls sie noch nicht geöffnet sind. Kann **relate4retrieve** eine in der Relationsdatei angegebene Datendatei nicht finden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus der Relation ausgeklammert und der Versuch unternommen, die nächste Datendatei zu lokalisieren.

Hat *openFiles* einen falschen Wert (0), geht **relate4retrieve** davon aus, daß alle Daten-, Index- und Memodateien bereits geöffnet sind. Kann eine in der Relationsdatei angegebene Datendatei nicht geöffnet werden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus der Relation ausgeklammert, und **relate4-retrieve** setzt den Aufbau der Relation fort.

dataPathName — Bei diesem Parameter handelt es sich um eine Zeichenkette, die für die in der Relation gespeicherten Daten-, Index- und Memodateien einen neuen Pfadnamen angibt.

Ist *dataPathName* ein Nullstring (""), werden die in der Relationsdatei abgelegten Pfade verwendet. Enthält die Datei keine Pfadangaben, versucht **relate4retrieve**, die Dateien im aktuellen Verzeichnis zu öffnen.

Wenn Sie *dataPathName* angeben, werden dadurch die in der Relationsdatei gespeicherten Pfade außer Kraft gesetzt.

Rückgaben

Nicht-Null	Die Relation wurde erfolgreich aus der angegebenen Relationsdatei eingelesen.
Null	Beim Lesen der Relationsdatei oder beim Öffnen der Top-Master-Datendatei hat sich ein Fehler ereignet. Fragen Sie **CODE4.errorCode** ab, um den genauen Fehler zu ermitteln.

Siehe auch **relate4save, relate4init, relate4free**

relate4save

Aufruf rc = relate4save(RELATE4&, fileName$, savePathNames%)

Beschreibung Mit dieser Funktion wird die angegebene Relation in einer Relationsdatei gespeichert.

Parameter	RELATE4	Ein Zeiger auf die Relation, die in einer Relationsdatei gespeichert werden soll.
	fileName	Eine Zeichenkette, die den Namen (einschließlich Laufwerk und Verzeichnis) der Relationsdatei enthält. Da stets die Erweiterung .REL verwendet wird, brauchen Sie keinen Dateityp einzugeben.
	savePathNames	Wenn dieser Parameter einen wahren Wert (1) enthält, speichert **relate4save** die in der Relation verwendeten Dateinamen als vollständige Pfadnamen ab. Ist der Wert von *savePathNames* falsch (0), wird lediglich der reine Dateiname gespeichert.
Rückgaben	0	Relationsdatei erfolgreich gespeichert.
	r4noCreate	Die Relationsdatei konnte nicht angelegt werden. Die Ursache dafür besteht im allgemeinen darin, daß *fileName* bereits vorhanden ist oder die Anwendung keine Lese-/Schreibberechtigung für das angegebene Laufwerk besitzt.
	< 0	Fehler.

Siehe auch **relate4retrieve**

relate4topMaster

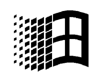
Aufruf RELATE4& = relate4topMaster(RELATE4&)

Beschreibung Diese Funktion gibt einen Zeiger auf die **RELATE4**-Struktur des Top Masters im Relationsset zurück.

Parameter RELATE4 Ein Zeiger auf eine beliebige **RELATE4**-Struktur im Relationsset.

Rückgaben	Nicht-Null	Ein Zeiger auf die **RELATE4**-Struktur der Top-Master-Relation.
	0	Fehler.

Siehe auch **relate4master** im CodeBasic-5-Referenzhandbuch



report4caption

Aufruf rc% = report4caption(REPORT4&, caption$)

Beschreibung Mit dieser Funktion wird der Text in der Titelleiste des Reportausgabefensters bei der Bildschirmausgabe verändert.

Parameter	REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den die Titelleiste eingerichtet wird.
	caption	Eine Zeichenkette, die den Text enthält, der in der Titelleiste des Ausgabefensters erscheinen soll. **report4caption** legt eine Kopie von *caption* an.

Rückgaben	0	Der Reportkopf wurde erfolgreich eingerichtet.
	< 0	Fehler.

report4currency

Aufruf rc% = report4currency(REPORT4&, currency$)

Beschreibung Mit dieser Funktion wird der Text festgelegt, der unmittelbar links von solchen numerischen Ausgabeobjekten erscheinen soll, die als Währungswerte formatiert sind.

Parameter	REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den das Währungssymbol definiert wird.

	currency	Eine Zeichenkette, die das Währungssymbol enthält. *currency* kann aus bis zu zehn Zeichen bestehen. **report4currency** legt eine Kopie von *currency* an. Wird diese Funktion nicht aufgerufen, geht das Programm vom Dollarzeichen ($) aus.
Rückgaben	0	Währungssymbol erfolgreich eingerichtet.
	< 0	Fehler.

report4dateFormat

Aufruf rc% = report4dateFormat(REPORT4&, format$)

Beschreibung Diese Funktion richtet das Standard-Datumsformat für den angegebenen Report ein. Alle neuen Ausgabeobjekte, die als Ergebnis einen Datumswert haben, benutzen dieses Format bei der Ausgabe. Beim ersten Anlegen des Reports wird der Wert des Elements **CODE4.dateFormat** als Standard-Datumsformat des Reports gespeichert.

Parameter	REPORT4	Ein Zeiger auf den Report, für den das Datumsformat eingestellt wird.
	format	Eine Zeichenkette, die das Standard-Datumsformat enthält. Diese Zeichenkette sollte aus den Schablonenformatierzeichen bestehen („D“, „M“, „C“, „Y“). **report4dateFormat** legt eine Kopie von *format* an.
Rückgaben	0	Erfolg.
	< 0	Fehler. *REPORT4* ist ungültig.

report4decimal

Aufruf rc% = report4decimal(REPORT4&, decimalChar$)

Beschreibung Mit dieser Funktion wird das Zeichen bestimmt, das bei numerischen Ausgabeobjekten die ganzen Zahlen von den Bruchwerten trennt.

Parameter

REPORT4	Ein **REPORT4**-Zeiger auf den Report, bei dem das Dezimalzeichen verwendet wird.
decimalChar	Das als Dezimalzeichen zu verwendende Zeichen. Voreingestellt ist der Dezimalpunkt (.).

Rückgaben

0	Erfolg.
< 0	Fehler. *REPORT4* ist ungültig.

report4do

Aufruf rc% = report4do(REPORT4&)

Beschreibung Mit dieser Funktion wird der Report auf das ausgewählte Gerät ausgegeben.

Bei der Ausgabe unter Windows deaktiviert **report4do** das (mit **report4parent** bezeichnete) Elternfenster des Reports, bis das Reportfenster geschlossen wird. Auf diese Weise wird die Anwendung daran gehindert, während der Ausführung reportrelevante Daten zu aktualisieren.

Achtung! Bei Windows-Programmen müssen Sie vor dem Aufruf dieser Funktion zunächst **report4parent** aufrufen. Erfolgt dieser Aufruf nicht, kann das zu unvorhersehbaren Ergebnissen führen.

Parameter

REPORT4	Bezeichnet den auszugebenden Report.

Rückgaben	0	Der Report wurde erfolgreich ausgegeben.
	r4terminate	Es konnte keine Relation hergestellt werden, und bei der mit **relate4errorAction** angegebenen Fehlerbehandlung handelte es sich um **relate4terminate**.
	< 0	Fehler.

Siehe auch **report4toScreen, report4printerSelect, report4output**

report4free

Aufruf Call report4free(REPORT4&, freeRelate%, closeFiles%)

Beschreibung Dieser Befehl gibt den gesamten mit dem Report verbundenen Speicher frei.

Parameter	REPORT4	Ein **REPORT4**-Zeiger auf den Report, der aus dem Speicher gelöscht werden soll,
	freeRelate	Hat dieser Parameter einen wahren Wert (1), wird der mit der Relation des Reports verbundene Speicher automatisch freigegeben. Wird ein falscher Wert (0) übergeben, wirkt sich das nicht auf die Relation aus.
	closeFiles	Dieser Parameter schließt, wenn er einen wahren Wert (1) enthält, automatisch die im Report genannten Daten-, Index- und Memodateien. Sind *closeFiles* oder *freeRelate* falsch, wird die Einstellung ignoriert.

Siehe auch **relate4free** im CodeBasic-5-Referenzhandbuch

report4margins

Aufruf rc% = report4margins(REPORT4&, left&, right&, top&, bottom&, unitType%)

Beschreibung Mit dieser Funktion werden die Standard-Randeinstellungen des Reports verändert.

Hinweis: Einige Ausgabegeräte, z. B. Laserdrucker, haben einen hardwaremäßig eingestellten Randbereich, der nicht bedruckbar ist. **report4margins** prüft diese Tatsache ab und läßt es nicht zu, daß die physikalischen Ränder des Geräts nicht eingehalten werden.

Parameter

REPORT4 Ein **REPORT4**-Zeiger auf den Report, bei dem die Ränder eingestellt werden.

left Breite des linken Randes in den Schritten der eingestellten Maßeinheit.

right Breite des rechten Randes in den Schritten der eingestellten Maßeinheit.

top Breite des oberen Randes in den Schritten der eingestellten Maßeinheit.

bottom Breite des unteren Randes in den Schritten der eingestellten Maßeinheit.

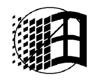
unitType Maßeinheit für die oben genannten Randeinstellungen. Bei graphischen Benutzerschnittstellen lassen sich dabei recht gut Angaben in tausendstel Zoll verwenden. Bei zeichenbezogenen Schnittstellen eignen sich Angaben in Zeichen oft besser. *unitType* kann folgende Werte annehmen:

1 Die Maßeinheiten werden in Zeichen angegeben.

0 Die Maßeinheiten werden in tausendstel Zoll angegeben.

Rückgaben 0 Die Ränder wurden erfolgreich eingestellt.
< 0 Fehler.

Siehe auch **report4pageSize**



report4output

Aufruf rc% = report4output(REPORT4&, outputHandle%, useStyles%)

Beschreibung Der Aufruf dieser Funktion erfolgt vor der Erzeugung des Reports und weist **report4do** an, den Report auf ein System-Handle wie „standard out", „standard print" oder in eine offene Datei zu schreiben. Wird diese Funktion nicht aufgerufen, erfolgt die Reportausgabe über **report4do** auf dem „standard out"-Bildschirm.

Diese Funktion wird auch eingesetzt, wenn ein Report in eine Datei ausgegeben wird.

Parameter

REPORT4 Der Report, für den ein Ausgabeziel eingerichtet wird.

outputHandle Ein Standard-System-Handle, wie es von der BASIC-Funktion **FILEATTR** zurückgegeben wird. Andere vordefinierte Handles sind:

1 „standard out"; per Voreinstellung ist das der Bildschirm.

4 „standard print". Bei PCs handelt es sich dabei gewöhnlich um den Drucker am Anschluß LPT1.

useStyles Mit diesem Parameter wird festgelegt, ob die von **report4do** ausgegebenen Daten die Druckersteuersequenzen, die in den Seiten-Layouts definiert sind, enthalten sollen oder nicht. *useStyles* muß einen der folgenden Werte annehmen:

1 Die Ein- und Ausleitungssequenzen des Drukkers werden in das angegebene Handle ausgegeben.

0 Die Druckersteuersequenzen werden ignoriert, und es wird lediglich der Text für die Ausgabeobjekte ausgegeben.

Rückgaben

0	Einstellungen erfolgreich vorgenommen.
< 0	Fehler. *REPORT4* ist ungültig.

Siehe auch **report4do**

report4pageSize

Aufruf rc% = report4pageSize(REPORT4&, height&, width&, unitType%)

Beschreibung Mit dieser Funktion werden Breite und Höhe der Reportseite eingestellt. Bei einer Windows-Anwendung wird das aktuelle Seitenformat des ausgewählten Druckers verwendet. Bei einer Non-Windows-Anwendung wird das voreingestellte Seitenformat von 25 x 80 Zeichen verwendet.

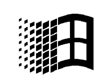
Parameter

REPORT4	Ein Zeiger auf den Report, für den das Seitenformat eingestellt wird.
height	Die Höhe der Ausgabeseite in der angegebenen Maßeinheit.
width	Die Breite der Ausgabeseite in der angegebenen Maßeinheit.
unitType	Mit diesem Parameter wird die Maßeinheit von *height* und *width* festgelegt. *unitType* kann einen der folgenden Werte annehmen:

1 *height* und *width* werden in Zeichen angegeben.

	0	*height* und *width* werden in tausendstel Zoll angegeben.
Rückgaben	0	Das Seitenformat wurde erfolgreich eingerichtet.
	< 0	Fehler.

Siehe auch **report4margins, report4printerSelect**



report4parent

Aufruf rc% = report4parent(REPORT4&, Form.hWnd%)

Beschreibung Diese Funktion legt das für die Reportausgabe zu verwendende Elternfenster fest. *Form.hWnd* sollte die **.hWnd**-Eigenschaft des Formulars in Ihrer Visual-Basic-Anwendung sein, an das der Fokus nach dem Schließen des Reportfensters zurückgegeben wird.

Parameter	REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den das Elternfenster eingerichtet wird.
	Form.hWnd	Ein Microsoft-Windows-Handle auf das Formular, dessen Fokus nach der Reportausgabe eingestellt wird. Wird die Ausgabe an ein Fenster geschickt, wird dieses Formular bis zum Schließen des Reports deaktiviert.
Rückgaben	0	Erfolg.
	< 0	Fehler. *REPORT4* ist ungültig.

Achtung! Bei Windows-Anwendungen muß diese Funktion vor **report4do** aufgerufen werden. Geschieht das nicht, kann das zu unvorhersehbaren Ergebnissen führen.

report4printerSelect

Aufruf Call report4printerSelect(REPORT4&)

Beschreibung Dieser Befehl ruft die Dialogbox „Druckereinrichtung“ auf, um einen Drucker für den Report auszuwählen.

Damit die Funktion einwandfrei arbeitet, muß die dynamische Link-Bibliothek COMMDLG.DLL von Microsoft Windows 3.1 vorhanden sein. Die Datei wird gewöhnlich vom Windows-Setup im Unterverzeichnis .\WINDOWS\SYSTEM installiert. Läuft Ihre Anwendung unter Windows 3.0 oder fehlt die Datei in dem Verzeichnis, können Sie diese Funktion nicht ausführen.

Parameter REPORT4 Ein **REPORT4**-Zeiger auf den Report, der für den ausgewählten Drucker konfiguriert ist.

report4querySet

Aufruf rc% = report4querySet(REPORT4&, queryExpr$)

Beschreibung Diese Funktion stellt eine Abfrage für das Relationsset ein. Dabei wird der Ausdruck *queryExpr* für jeden zusammengesetzten Datensatz ausgewertet. Ist der Ausdruck wahr, geht der Datensatz in den Report ein. Ist das Ergebnis von *queryExpr* falsch, wird der Datensatz ignoriert.

Parameter REPORT4 Ein **REPORT4**-Zeiger auf den Report, für den die Abfrage eingerichtet wird.

queryExpr Ein logischer dBASE-Ausdruck, der Datensätze der zusammengesetzten Datendatei filtert. Ist *queryExpr* ein Nullstring (""), werden alle Datensätze der zusammengesetzten Datendatei im Report verwendet.

Hinweis: Im Abfrageausdruck vorhandene Feldnamen müssen mit Datendateikennzeichner verwendet werden. So ist `"DBF->NAME = 'SCHMIDT'"` z. B. ein gültiger Abfrageausdruck.

Rückgaben

0	Die Abfrage wurde erfolgreich eingerichtet.
< 0	Fehler.

Siehe auch **relate4querySet, relate4sortSet, report4sortSet**

report4relate

Aufruf RELATE4& = report4relate(REPORT4&)

Beschreibung Diese Funktion gibt einen Zeiger auf die mit dem Report verbundene **RELATE4**-Struktur zurück.

Parameter

REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den der **RELATE4**-Zeiger zurückgegeben wird.

Rückgaben

Nicht-Null	Der mit dem Report verbundene **RELATE4**-Zeiger.
0	Fehler.

Siehe auch **report4querySet, report4sortSet**

report4retrieve

Aufruf REPORT4& = report4retrieve(CODE4&, fileName$, openFiles%, dataPath$)

Beschreibung Diese Funktion liest eine Reportdatei von der Platte ein und baut die entsprechende **REPORT4**-Struktur auf. Parallel dazu wird im Hintergrund ein Relationsset mit der entsprechenden **RELATE4**-Struktur erzeugt.

Parameter	CODE4	Ein Zeiger auf die **CODE4**-Struktur der Anwendung; er dient der Speicherverwaltung und der Fehlerbehandlung.
	fileName	Eine Zeichenkette, die Laufwerk, Verzeichnis und Namen der Reportdatei enthält. Geben Sie keine Erweiterung am Dateinamen an, geht **report4retrieve** von der Erweiterung .REP aus.
	openFiles	Hat *openFiles* einen wahren Wert (1), versucht **report4retrieve**, die im Report genannten Datendateien zu öffnen, falls sie noch nicht geöffnet sind. Kann eine angegebene Datendatei nicht lokalisiert werden, wird diese Datei sowie alle ihr zugeordneten Slavedateien aus dem Report ausgeklammert. Alle Ausgabeobjekte und/oder Ausdrücke, die sich auf die fehlenden Datendateien beziehen, werden automatisch aus dem Report entfernt. Hat *openFiles* einen falschen Wert (0), geht das Programm davon aus, daß alle Dateien bereits geöffnet sind.
	dataPath	Ist *dataPath* ein Nullstring (""), benutzt **report4retrieve** zur Lokalisierung der Datendateien die in der Reportdatei gespeicherten Pfade. Wenn die Datei keine Pfadnamen enthält, geht **report4retrieve** davon aus, daß sich die Datendateien im aktuellen Verzeichnis befinden. Ist *dataPath* nicht Null, enthält es als Zeichenkette Laufwerk und/oder Verzeichnis, in dem sich die Datendateien des Reports befinden. Das Verzeichnis *dataPath* setzt alle innerhalb des Reports gespeicherten Pfade außer Kraft.
Rückgaben	Nicht-Null	Der Report wurde erfolgreich geladen. Den zurückgegebenen **REPORT4**-Zeiger können auch andere Funktionen des Reportmoduls benutzen.

0	Fehler. Der Report konnte nicht geladen werden. Das liegt möglicherweise daran, daß die Top-Master-Datendatei nicht lokalisiert oder nicht genug Speicher für den Report zugewiesen werden konnte.

Siehe auch **relate4retrieve**

report4save

Aufruf rc% = report4save(REPORT4&, fileName$, savePaths%)

Beschreibung Diese Funktion legt einen Report in einer variabel codierten Reportdatei ab, die entweder mit CodeReporter bzw. der Funktion **report4retrieve** wieder eingelesen werden kann.

Hinweis: **report4save** wirkt sich in keinster Weise auf den im Speicher befindlichen Report aus. Die Funktion kann ohne schädliche Wirkung vor oder nach **report4do** aufgerufen werden.

Achtung! Wird ein Report mit Grafikausgabeobjekten in eine Non-Windows-Anwendung geladen und mit **report4save** abgespeichert, werden die Grafikausgabeobjekte in der neuen Reportdatei nicht mitgesichert.

Parameter

REPORT4	Ein **REPORT4**-Zeiger auf den Report, der auf der Platte abgelegt werden soll.
fileName	Eine Zeichenkette, die Laufwerk, Verzeichnis und Namen der Datei enthält, unter denen der Report gespeichert werden soll. Sie können eine Erweiterung am Dateinamen angeben; geben Sie keine an, wird die voreingestellte Erweiterung .REP an den Dateinamen angehängt.
savePaths	Enthält *savePaths* einen wahren Wert (1), legt **report4save** für jede im Report genannte Datei

Laufwerk und Pfad in der Reportdatei ab. Enthält *savePaths* einen falschen Wert (0), werden lediglich die im Report vorhandenen Dateinamen gespeichert.

Rückgaben

0	Der Report wurde erfolgreich in die angegebene Datei geschrieben.
< 0	Fehler. *REPORT4* ist ungültig, oder die angegebene Datei konnte nicht angelegt werden.

Siehe auch **report4retrieve**

report4separator

Aufruf rc% = report4separator(REPORT4&, separator$)

Beschreibung Mit dieser Funktion wird das Zeichen angegeben, das zwischen Hundertern und Tausendern, Tausendern und Millionen usw. steht.

Parameter

REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den das Zifferntrennzeichen gelten soll.
separator	Das als Trennzeichen verwendete Zeichen. Wünschen Sie kein Trennzeichen, übergeben Sie für *separator* einen Nullstring (""). Wird diese Funktion nicht aufgerufen, wird als voreingestelltes Trennzeichen das Komma (,) benutzt.

Rückgaben

0	Trennzeichen erfolgreich eingerichtet.
< 0	Fehler. *REPORT4* ist ungültig.

report4sortSet

Aufruf rc% = report4sortSet(REPORT4&, sortExpr$)

Beschreibung Diese Funktion legt die Sortierfolge fest, in der die zusammengesetzten Datensätze des Reports eingelesen werden.

Parameter

REPORT4	Ein **REPORT4**-Zeiger auf den Report, für den die Sortierfolge gelten soll.
sortExpr	Eine Zeichenkette, die den dBASE-Ausdruck enthält, nach dem die zusammengesetzte Datendatei sortiert wird. Dieser Ausdruck kann als Ergebnis einen Zeichen-, Datums- oder einen numerischen Wert haben.

Hinweis: Im Sortierausdruck vorhandene Feldnamen müssen mit Datendateikennzeichner verwendet werden. So ist `"DBF->NAME"` z. B. ein gültiger Sortierausdruck.

Rückgaben

0	Erfolg.
< 0	Fehler, oder *REPORT4* ist ungültig.

Siehe auch **relate4sortSet, report4querySet**

report4toScreen

Aufruf rc% = report4toScreen(REPORT4&, toScreen%)

Beschreibung Mit dieser Funktion wird **report4do** angewiesen, ein Fenster zu erzeugen und die Reportausgabe darin vorzunehmen oder statt dessen den Report auf den ausgewählten Drucker zu schicken.

Per Voreinstellung erfolgt die Reportausgabe mit **report4do** in einem Fenster.

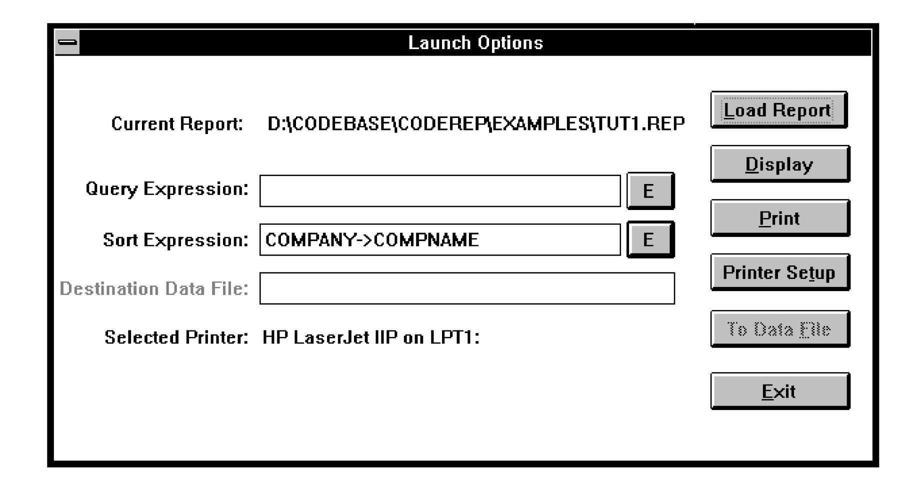
Parameter	REPORT4	Ein **REPORT4**-Zeiger auf den Report, der auf dem Bildschirm ausgegeben werden soll.
	toScreen	Mit diesem Parameter wird das Ausgabeziel des Reports festgelegt. *toScreen* kann einen der folgenden Werte annehmen: 1 Es wird ein Fenster geöffnet, und die Reportausgabe erfolgt in dem Fenster. 0 Die Reportausgabe erfolgt auf dem ausgewählten Drucker.
Rückgaben	>= 0	Die vorhergehende Einstellung von *toScreen* wird zurückgegeben.
	< 0	Fehler. *REPORT4* oder *toScreen* sind ungültig.
Siehe auch	**report4output, report4do**	

Anhang G: Launch-Programme

Im Lieferumfang von CodeReporter sind Dienstprogramme enthalten, mit deren Hilfe Reportdateien auch außerhalb von CodeReporter ausgegeben werden können. Diese Dienstprogramme beschreiben wir im folgenden.

Windows

CodeReporter stellt Ihnen zur Ausgabe von Reportdateien das Windows-Programm LAUNCH_W.EXE zur Verfügung. Dabei handelt es sich um ein ausführbares Programm, mit dem Sie direkt aus Windows Reports einsehen oder drucken können, ohne vorher CodeReporter aufzurufen. LAUNCH_W verfügt sowohl über eine interaktive Schnittstelle als auch über Befehlszeilenparameter, die in Symbolen verwendet werden können.



Um die Indexdatei-Kompatibilität bestimmen zu können, benötigt LAUNCH_W.EXE die CodeReporter-DLL CR2CODE.DLL. Einzelheiten dazu entnehmen Sie bitte dem Kapitel „Starten mit CodeReporter“.

Als weiteres Beispiel für den Einsatz von CodeReporter-Reportdateien sowie für die Reportfunktionen unter Windows ist im Lieferumfang ebenfalls der Quellcode von LAUNCH_W.C enthalten.

Wird LAUNCH_W ohne Befehlszeilenparameter ausgeführt, wird die Dialogbox „Launch Options“ aufgerufen, bei dem lediglich die Schaltflächen „Load Report“ und „Exit“ angeklickt werden können. Klicken Sie auf „Load Report“, erscheint die Dialogbox „Specify Report“, aus der Sie eine Reportdatei auswählen können. Klicken Sie auf „Exit“, wird das Programm beendet.

Über die Dialogbox „Launch Options“ bestimmen Sie die Zieldatei des Reports und definieren Sortier- und/oder Abfrageausdrücke.

Wenn Sie den aktuellen Report aus Versehen geladen haben bzw. wenn ein anderer Report ausgegeben werden soll, rufen Sie über die Schaltfläche „Load Report“ die Dialogbox „Select Report“ auf. Mit Hilfe dieser Dialogbox können Sie eine neue Reportdatei lokalisieren und öffnen. Ist die Dialogbox „Launch Options“ wieder auf dem Bildschirm zu sehen, wird der neue Report geladen.

Klicken Sie auf die Schaltfläche „Display“, wird der Report in einem Fenster ausgegeben. Das Seitenformat der Fensteranzeige entspricht dem des aktuellen Druckers, so daß Sie sich den Rest der Seite möglicherweise mit den Bildlaufleisten anzeigen lassen müssen.

Klicken Sie auf die Schaltfläche „Print“, wird der Report über den aktuellen Drucker ausgegeben. Während des Druckvorgangs teilt Ihnen eine Dialogbox eben diesen Sachverhalt mit. Ein Klick auf die Schaltfläche „Cancel“ bricht den Druckvorgang ab.

Soll der Report nicht über den Standard-Drucker von Windows ausgegeben werden oder müssen Sie die Druckereinstellungen verändern, können Sie die Dialogbox „Druckereinrichtung“ über die Schaltfläche „Printer Setup“ aufrufen. In dieser Dialogbox können Sie den Ausgabedrucker bestimmen und ihn konfigurieren.

Wenn die Reportdatei eine (mit **REPORT | VORGABEN AUSGABEDATEI** erstellte) Schablone für die Ausgabedatei enthält, können Sie die Schaltfläche „To Data File“ anklicken. Dabei wird der Report in die Datendatei ausgegeben, die Sie ins Editierfeld „Ausgabe-Datendatei“ eingetragen haben.

Enthält der Report keine Schablone für die Ausgabedatei, ist die Schaltfläche „To Data File“ von vornherein deaktiviert, und der Report läßt sich nicht in eine Datendatei ausgeben.

Mit dem Launch-Programm können Sie auch die im Report gespeicherten Sortier- und Abfrageausdrücke abändern. Nehmen Sie diese Änderungen in den Editierfeldern „Sort Expression“ und/oder „Query Expression“ vor, spiegelt die Reportausgabe die neuen Einstellungen wider.

Weitere Einzelheiten zum Sortieren und Abfragen eines Reports entnehmen Sie bitte dem Kapitel „Relationale Reports“.

Nach der Reportausgabe kehrt das Launch-Programm zu dieser Dialogbox zurück.

Befehlszeile

LAUNCH_W gestattet die Eingabe von Befehlszeilenparametern, die den Lade- und Anzeigevorgang automatisieren. Die Parameter können Sie zu den Programmeigenschaften von LAUNCH_W hinzufügen oder im Programm-Manager als Option unter Datei | Ausführen eingeben. Einzelheiten zur Verwendung von Befehlszeilenparametern bei Windows-Anwendungen finden Sie in Ihrem *Windows-Benutzerhandbuch*.

Aufruf LAUNCH_W [{*name*} [-q{*expr*}] [-s{*expr*}]
[[-v | -p] | [{ -d{*name*}} [-q{*expr*}][-s{*expr*}]] [. . .]]]

name — Der Name des ersten Reports, der ausgegeben oder geladen werden soll.

-d{name} — Sollen weitere Reports ausgegeben werden, können Sie diese unter der Option *-d* angeben. Sie dient gleichsam als Trennzeichen zwischen den Reports.

-q{expr} — Voreingestellten Abfrageausdruck ändern. Die Abfrageausdrücke müssen in doppelte Anführungszeichen (") eingeschlossen werden, zum Beispiel:

```
LAUNCH_W SAMPLE.REP -q"DBF->FELDNAME > 'LINCOLN'"
```

Zeichenliterale, wie z. B. „Lincoln“ oben, müssen in einfache Anführungszeichen (') eingeschlossen werden.

-s{expr} — Voreingestellten Sortierausdruck ändern. Sortierausdrücke müssen in doppelte Anführungszeichen (") eingeschlossen werden, zum Beispiel:

```
LAUNCH_W SAMPLE.REP -s"DBF->FELDNAME"
```

Auch hierbei müssen Zeichenliterale in einfache Anführungszeichen (') eingeschlossen werden.

-v/-p Die Option -*v* zeigt den angegebenen Report automatisch auf dem Bildschirm an. Mit der Option -*p* wird der Report automatisch auf dem Standard-Drucker von Windows ausgegeben. Geben Sie diese Parameter ein, wird LAUNCH_W als Symbol aufgerufen.

Wenn Sie keine der beiden Optionen verwenden, wird der angegebene Report geladen und die Dialogbox „Launch Options“ aufgerufen; es wird lediglich der erste Report geladen.

-*p* und -*v* dürfen nicht gleichzeitig aufgerufen werden. Ihr Aufruf gilt für alle Reports, die geladen werden.

Non-Windows

CodeReporter stellt Ihnen vier indexspezifische Non-Windows-Programme zur Ausgabe von CodeReporter-Reportdateien zur Verfügung. Bei diesen Programmen handelt es sich ausführbare DOS-Dateien, mit denen Sie sich Reports von der DOS-Befehlszeile aus oder über Stapelverarbeitungsdateien anzeigen und ausdrucken lassen können.

Als weiteres Beispiel für den Einsatz der Reportfunktionen und als Möglichkeit, LAUNCH_D unter anderen Betriebssystemen einzusetzen, ist im Lieferumfang ebenfalls der Quellcode von LAUNCH_D enthalten.

Hinweis: Das Non-Windows-Launch-Programm kann nicht auf Windows-spezifische Styles zugreifen. Aus diesem Grund unterscheiden sich Reports, die mit diesem Programm ausgegeben werden, von Reports, die unter CodeReporter ausgegeben werden.

Die Größen von Gruppenkopf- und -fuß-Bereichen – wie auch Titel-, Schluß-, Seitenkopf- und -fuß-Bereich – werden auf 0,42 cm (1/6", etwa 12 Pkt.) gerundet, da bei den meisten Druckern eine Zeile 0,42 cm hoch ist.

Dateinamen

Das Launch-Programm verhält sich indexspezifisch. Um zum Beispiel einen Report mit FoxPro-Indexdateien auszuführen, müssen Sie eine FoxPro-kompatible Version des Launch-Programms aufrufen. Die vorkompilierten Versionen des Launch-Programms (im Verzeichnis .\LAUNCH installiert) heißen folgendermaßen:

Name	Kompatibilität
LNCH_FOX	FoxPro 2.0 (oder höher) (.CDX)
LNCH_MDX	dBASE IV (nur .MDX)
LNCH_CLI	Clipper (.NTX)
LNCH_NDX	dBASE III (.NDX)

Im vorliegenden Handbuch bezeichnen wir das Launch-Programm per Übereinkunft als LAUNCH_D.

Aufruf LAUNCH_D {name} [-q{*expr*}] [-s{*expr*}] [-x{*nn*}] [-y{*nn*}] [-p[*dest*] | -f[*dataName*] [-t]

-q{expr} Voreingestellten Abfrageausdruck ändern. Die Abfrageausdrücke müssen in doppelte Anführungszeichen (") eingeschlossen werden, zum Beispiel:

```
LAUNCH_D SAMPLE.REP -q"DBF->FELDNAME > 'LINCOLN'"
```

Zeichenliterale, wie z. B. „Lincoln“ oben, müssen in einfache Anführungszeichen (') eingeschlossen werden.

-s{expr} Voreingestellten Sortierausdruck ändern. Sortierausdrücke müssen in doppelte Anführungszeichen (") eingeschlossen werden, zum Beispiel:

```
LAUNCH_D SAMPLE.REP -s"DBF->FELDNAME"
```

Auch hierbei müssen Zeichenliterale in einfache Anführungszeichen (') eingeschlossen werden.

-x{nn} Voreingestellte Breite der Seite (in Zeichen) ändern. Als Standard sind 80 Zeichen pro Zeile eingestellt. Zum Beispiel:

```
LAUNCH_D SAMPLE.REP -x70
```

-y{nn} Voreingestellte Höhe der Seite (in Zeilen) ändern. Als Standard sind 25 Zeilen eingestellt. Die meisten Drucker bringen 66 Zeilen auf einer Seite unter. Zum Beispiel:

```
LAUNCH_D SAMPLE.REP -y66
```

-p{dest} Voreingestelltes Ausgabeziel verändern. Reports werden standardmäßig auf dem Bildschirm ausgegeben. Geben Sie nur *-p* an, erfolgt die Ausgabe auf „standard print“. Wenn Sie eine Zeichenkette angeben, erfolgt die Ausgabe in eine Datei mit dem angegebenen Namen. Zum Beispiel:

LAUNCH_D SAMPLE.REP -p <== Ausgabe auf „standard print“

LAUNCH_D SAMPLE.REP -pSAMPLE.OUT <== Ausgabe in Datei

LAUNCH_D SAMPLE.REP -pLPT2 <== Ausgabe auf Anschluß LPT2

Die Option *-p* kann in Verbindung mit *-f* verwendet werden.

-f[dataName] Die Ausgabe des Reports soll in eine Datendatei erfolgen. Diese Option ist lediglich möglich, wenn der Report mit einer Datendateischablone gespeichert worden ist. Geben Sie *dataName* nicht an, erfolgt die Ausgabe in die im Report angegebene Datendatei. Geben Sie *dataName* an, wird eine entsprechende Datendatei angelegt, und die Reportausgabe erfolgt in diese Datei.

-t Non-Windows-Druckercodes verwenden, die in den Styles des Reports vorhanden sind. Per Voreinstellung übermittelt LAUNCH_D die Druckersteuerzeichen nicht.

Index

CodeReporter License Agreement

This legal document is an agreement between you, the CodeReporter LICENSEE, and Sequiter Software Inc. (hereinafter referred to as the „Agreement“).

Sequiter Software Inc. is not „selling“ its software. Instead, Sequiter Software Inc. is granting the right to use its software through this license agreement. Sequiter Software Inc. retains all ownership of its software including all copies of its software.

Definitions „Software“ This is the Sequiter Software Inc. computer programs contained in this CodeReporter software package.

License Sequiter Software Inc. grants to you, and you hereby accept, a non-exclusive License to use the Software on a single computer at a single location.

You may install the Software on a file server provided that you ensure that no one other than yourself will be able to use the Software. Specifically, under this license, only one individual is allowed to use the Software. If more than one individual uses the Software, a license must be purchased for each individual.

Distribution

You may distribute the Report Launcher Executable Program. However, you may not distribute any other form of the Software except as expressly provided herein.

Transfer Restrictions The Software is licensed to you, and may not be transferred to anyone without the prior written consent of Sequiter Software Inc. Any authorized transferee of the Software shall be bound by the terms of this agreement. Regardless, you may not transfer, assign, rent, lease, sell or otherwise dispose of the Software except as expressly provided herein.

Disclaimer The Software and product items are provided „as is“ without any kind of warranty. It is your responsibility to determine whether the Software or product items are suitable for your purpose.

Miscellaneous This agreement is governed by the laws of the Province of Alberta, Canada. The LICENSEE consents to jurisdiction in the province of Alberta, Canada.